⑩

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

INVESTIGATION OF SPECTRAL-BASED TECHNIQUES
FOR CLASSIFICATION OF
WIDEBAND TRANSIENT SIGNALS IN
ADDITIVE WHITE GAUSSIAN NOISE

By
David A. DeRieux

March 1994

Thesis Advisor:	Dr. Ralph Hippenstiel
Co-Advisor:	Dr. Monique P. Fargues

Unclassified
Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report Approved for public release; distribution is unlimited.
2b Declassification/Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School — 6b Office Symbol *(If Applicable)* EC	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding/Sponsoring Organization — 8b Office Symbol *(If Applicable)*	9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*	10 Source of Funding Numbers Program Element Number — Project No — Task No — Work Unit Accession No

11 Title *(Include Security Classification)* INVESTIGATION OF SPECTRAL-BASED TECHNIQUES FOR CLASSIFICATION OF WIDEBAND TRANSIENT SIGNALS IN ADDITIVE WHITE GAUSSIAN NOISE

12 Personal Author(s) David A. DeRieux

13a Type of Report Master's Thesis	13b Time Covered From To	14 Date of Report *(year, month,day)* March 1994	15 Page Count 163

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	

19 Abstract *(continue on reverse if necessary and identify by block number*

Spectral-based classification schemes designed to separate various wide band transient signals in added noise have been identified and their performances compared along with those obtained using a back-propagation neural network implementation. The spectral-based measures used include: the normalized cross-correlation coefficient; the modified normalized cross-correlation coefficient, and; the divergence and the Bhattacharyya distance. Noise was added to the signals to create signal to noise ratios of 0 dB to -20 dB. Results show that as noise levels increase, the modified normalized cross-correlation coefficient spectral measure remains the most robust scheme.

20 Distribution/Availability of Abstract [✓ X] unclassified/unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Ralph Hippenstiel	22b Telephone *(Include Area code)* (408) 656-2633	22c Office Symbol EC/Hi

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted security classification of this page
All other editions are obsolete Unclassified

INVESTIGATION OF SPECTRAL-BASED TECHNIQUES FOR CLASSIFICATION OF WIDEBAND TRANSIENT SIGNALS IN ADDITIVE WHITE GAUSSIAN NOISE

by

David A. DeRieux
B.S.E.E., George Mason University, Alexandria VA, 1989

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

NAVAL POSTGRADUATE SCHOOL
March 1994

Author: [illegible]
David A. DeRieux

Approved by: Ralph Hippenstiel
Ralph Hippenstiel, Thesis Advisor

[illegible]
Monique P. Fargues, Co-Advisor

[illegible]
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

Spectral-based classification schemes designed to separate various wide band transient signals in added noise have been identified and their performances compared along with those obtained using a back-propagation neural network implementation. The spectral-based measures used include: the normalized cross-correlation coefficient; the modified normalized cross-correlation coefficient, and; the divergence and the Bhattacharyya distance. Noise was added to the signals to create signal to noise ratios of 0 dB to -20 dB. Results show that as noise levels increase, the modified normalized cross-correlation coefficient spectral measure remains the most robust scheme.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

In appreciation of their time, effort, and patience, I would like to thank my instructors, advisors, and the staff and faculty of Electrical & Computer Engineering Department, NPS. In addition, many thanks go to my fellow students with whom I spent long hours in class, in the office and in the library learning much more than what was taught in class.

Special thanks to my advisors Dr. Ralph Hippenstiel and Dr. Monique Fargues for the opportunity to do this research as well as their support, guidance, and encouragement as I learned.

Finally, I would like to thank my lovely wife, whose unwavering love and confidence were a constant source of strength and inspiration. Thank you Melinda.

I. INTRODUCTION

Classification of various wideband transient signals have extensibly been accomplished using neural networks (NN). Unfortunately, poor performance is attained if the neural network is trained with small training sets. In order to relieve training and computational time, four spectral measures using spectral coefficients are applied to correlate data sets supplied by the sponsor with a template derived from the classification of all data sets into three different groups of spectral coefficients. The goal of this research is to investigate the robustness of the spectral-based measures when white Gaussian noise is added to the transients under study.

First, all data sets are classified as in the original study and the correlation algorithms are verified [1]. White Gaussian noise is then added to each signal to create signal to noise ratios (SNR) of 0 dB to -20 dB. The resultant set of spectral coefficients is correlated with a user specified spectral template. Each spectral distance measure is calculated using one of four correlation algorithms. The results are plotted to determine which algorithm has the most robust performance as noise is added. Next, a NN implementation is used to determine its performance in a noisy environment. A tabular representation of the results is presented.

Chapter II details the spectral distance measures and correlation approaches, while Chapter III discusses related processing results. Next, Chapters IV and V cover the NN approach and related results. Finally, Chapter VI summarizes the work and addresses areas of further research.

II. SPECTRAL DISTANCE MEASURES AND SPECTRAL CORRELATION

A. PROCESSING OF DATA SETS

Typical transient signals are presented in Figure 2.1 and Figure 2.2. These signals are functions of time and are digitized for processing. The length of each signal is 1020 data points. Spectral coefficients can easily be obtained by taking the Fast Fourier Transform (FFT) of the digitized data and extracting the magnitude squared of the coefficients at the locations of interest once the signals have been zero-padded to lengths equal to 1024 points [1]. Plotting the spectral coefficients shows that the information is contained in the first 64 spectral locations.

Once the spectral coefficients are calculated and plotted, each is visually inspected and sorted into like sets. Similar transients in a train are grouped with other transients from either the same train or other trains. Figure 2.3 plots the frequency coefficient vs. power and frequency coefficient vs. signal location to further classify each transient. The spectral coefficients determined to belong to one group are averaged together to obtain a spectral template representing a particular signal group of interest. More robust results are obtained when a signal template is represented by many signals of the same group. Figure 2.4 represents a typical grouping of several sets of spectral coefficients into one set known as X1.

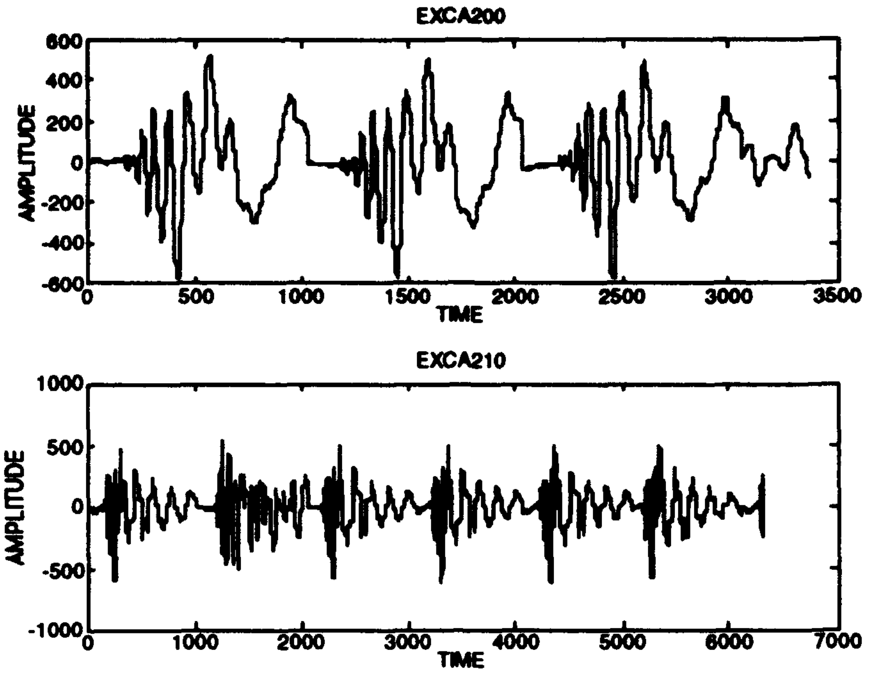


Figure 2.1: Typical Transient Type

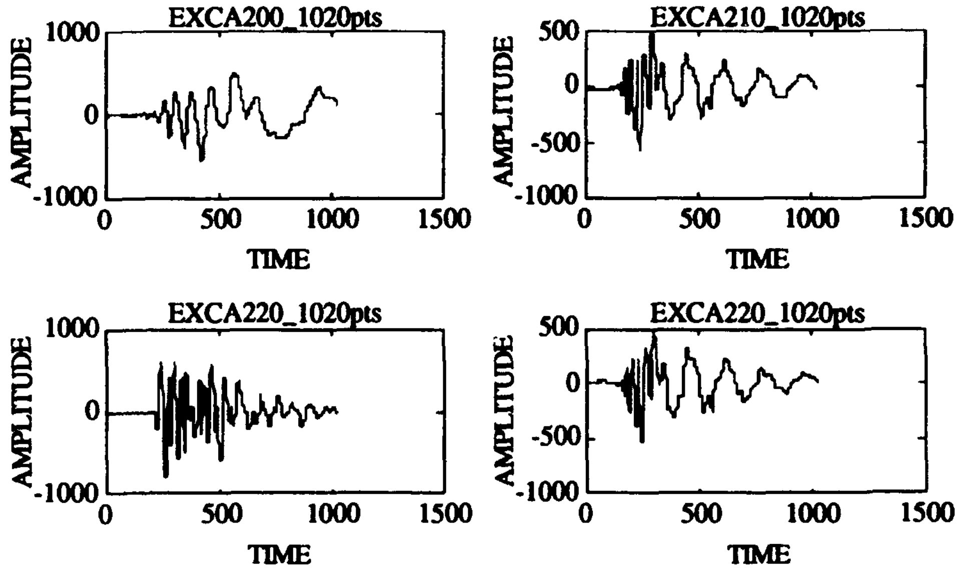


Figure 2.2: 1020 Point Transients

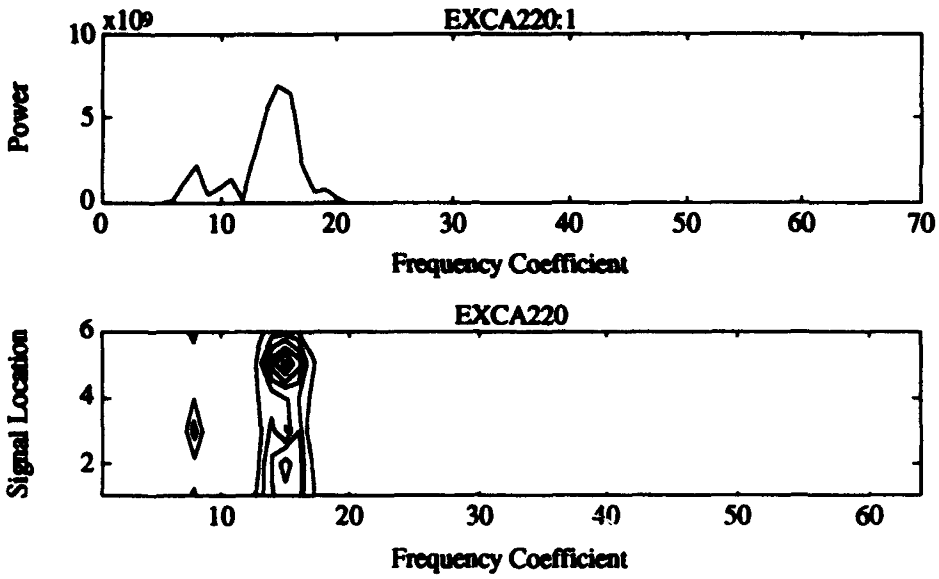


Figure 2.3: Spectral Plot for EXCA200:1 and Contour Plot for EXCA220

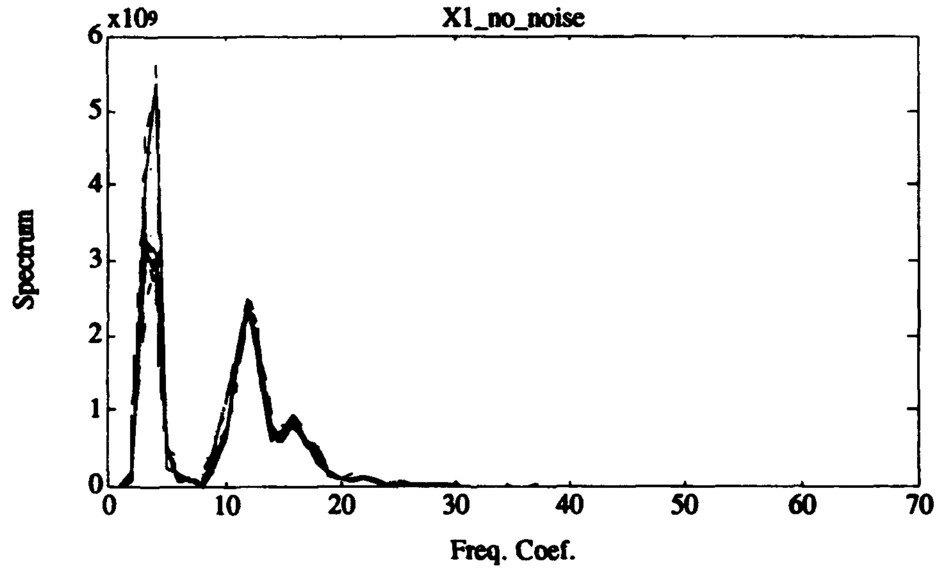


Figure 2.4: Superposition All Sets of Spectral Coefficients in X1

B. SPECTRAL MEASURES

Upon calculation of a template set, a decision is made whether or not the test signals belong to the set of spectral coefficients that created the template. This is determined by measuring the degree of similarity between the spectral density values. The four spectral measures chosen for the study are:

1.) Normalized cross-correlation coefficient,

2.) Modified cross-correlation coefficient,

3.) Bhattacharyya distance,

4.) Divergence.

With the spectral measures chosen, the first step in the signal processing sequence is to calculate the magnitude squared Fourier coefficients for the first 64 spectral locations. The magnitude squared form of the Fourier transform is given by the equation [1] [2]:

$$P_x(k) = \left| \sum_{n=0}^{1023} x(n) e^{-j2\pi \frac{kn}{1024}} \right|^2 \quad for \;\; k = 0,1 \ldots 63 \quad .$$

A rectangular window is employed in this study; however, it would be a simple modification to include any other data window. Some preliminary results using a Hamming data window did not show a significant difference with those obtained using the rectangular window. The inclusion of a window necessitates an additional 1020 point multiply per signal tested which increases the processing time. One open question is still the effect of windows when noise is present. The problem in general, is finding the degree of relatedness between the template and test vector and an appropriate threshold.

1. **Normalized Cross-Correlation Coefficient**

The first distance measure scheme implemented is the normalized cross-correlation coefficient. This approach involves cross correlating two vectors and normalizing the coefficients by the square root of the product of the auto powers. The output yields a value between zero and positive one. A value of one corresponds to a 100% match between two vectors, a value of zero corresponds to no match between the two vectors. The cross correlation expression is given by:

$$\rho_{XY} = \frac{\sum_{k=0}^{63} P_X(k) P_Y(k)}{\sqrt{\sum_{k=0}^{63} P_X^2(k) \sum_{k=0}^{63} P_Y^2(k)}} \quad .$$

Individual spectral measure results using the cross correlation coefficient are shown in the upper left-hand corner of Figures 2.5 and 2.6. Figure 2.5 shows a high degree of correlation between the X1 template and each set of spectral coefficients which make up that template. However, Figure 2.6 shows low correlation values, meaning that the X1 template is not related to the set X2.

2. **Modified Normalized Cross-Correlation Coefficient**

The modified normalized cross-correlation coefficient is virtually identical to the normalized cross-correlation coefficient with one important change. The mean value is removed from the template and test vector prior to correlation. This produces a sample mean of zero for each vector (i.e. DC component is zero). The range of values are intuitively pleasing: minus one to positive one. These ranges are preferable since distinguishing between sets is easier. For signals belonging to different sets, the correlation coefficients are typically smaller, while correlation coefficients of signals belonging to the same set generally remain the same. For automatic classification this method is both easier and more robust than other methods since the separation between unlike sets tends to be larger. The results for no-noise and for noisy environments also indicate this method of classification as the best technique. Typical correlation results are shown in the upper right-hand corner of Figures 2.5 and 2.6.

3. Bhattacharyya Distance [3]

Spectral densities can be converted into probability density functions (pdf) by normalizing the densities so that they have unit area. Information pertaining to the total power is lost, but all other information remains. Utilizing these pdfs, the Bhattacharyya distance can be applied. Defining $p_X()$ and $p_Y()$ as pdfs, the Bhattacharyya distance is given by:

$$B_{XY} = \int_{-\infty}^{\infty} \sqrt{p_X(f)\, p_Y(f)}\, df$$

for the continuous case, or

$$B_{XY} = \sum_{k=0}^{63} \sqrt{p_X(k)\, p_Y(k)}$$

for the discrete case.

If the two densities do not overlap, then B_{XY} is essentially zero. However, if the densities are identical, B_{XY} approaches one. Therefore, the Bhattacharyya Distance is a measure bounded by zero and one. Typical results are shown in the lower left-hand corner of Figures 2.5 and 2.6.

4. **Divergence (Kullback-Liebler Number) [3]**

This measure utilizes the pdf obtained from the normalized spectral densities. The divergence is defined as :

$$D_{XY}=\int_{-\infty}^{\infty}\left(\ln\frac{p_X(f)}{p_Y(f)}\right)p_X(f)df-\int_{-\infty}^{\infty}\left(\ln\frac{p_X(f)}{p_Y(f)}\right)p_Y(f)df$$

in the continuous case and

$$D_{XY}=\sum_{k=0}^{63}\left(\ln\frac{p_X(k)}{p_Y(k)}\right)p_X(k)-\sum_{k=0}^{63}\left(\ln\frac{p_X(k)}{p_Y(k)}\right)p_Y(k)$$

in the discrete case.

If $p_X(\ )$ equals $p_Y(\ )$, the divergence approaches zero. However, if the two density functions differ, then the divergence increases to become a large positive number. A determination of a usable threshold is difficult when using this measure. Typical processing of results are shown in the lower right-hand corner of Figures 2.5 and 2.6.

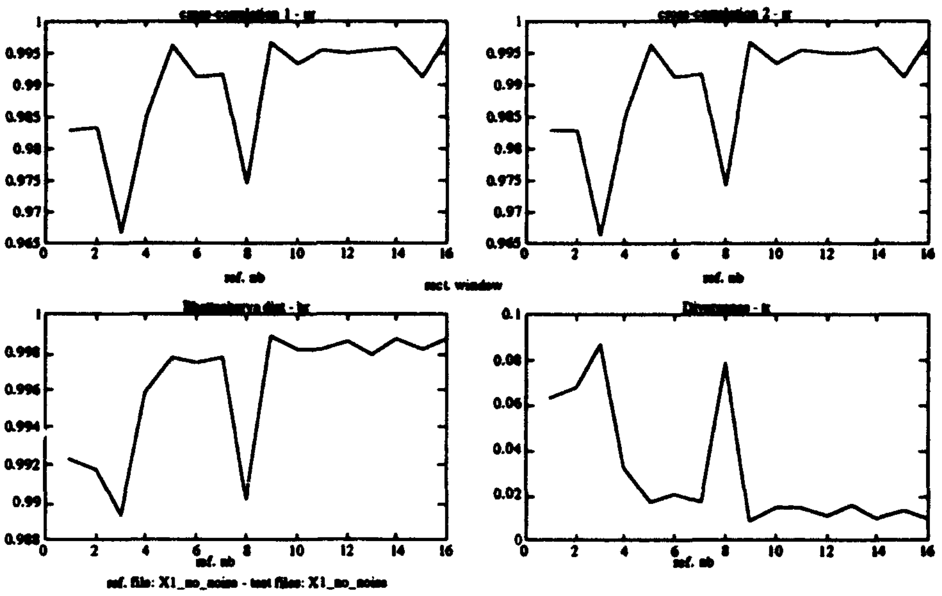


Figure 2.5: Spectral-Based Measures Using $X1_t$ and X1

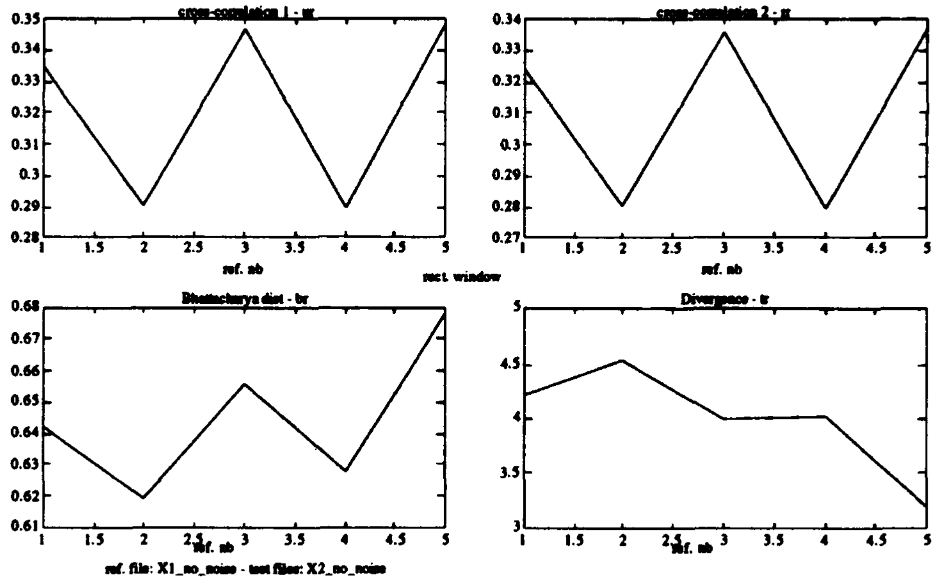


Figure 2.6: Spectral-Based Measures Using $X1_t$ and X2

III. SPECTRAL DISTANCE-BASED APPROACH

Four different transient signal sets are studied in this research. They are: EXCA200, EXCA210, EXCA220 and EXCA230.

As introduced in Chapter II, each signal set consists of several transients. Some transient sets are sampled at a slightly slower rate than others, thereby necessitating an operation to equalize all sampling rates to allow for comparison. The sampling frequency for set EXCA200 is 5.12 10^{10} Hz, while transient sets EXCA210, 220 and 230 are sampled at 2.048 10^{10} Hz. A resampled set of transients is produced when the original sampling rate does not conform with the one used for set EXCA200. To obtain SNR levels from 0 dB to -20 dB, noise is added to each transient within a set to test the performance of all four spectral measures as well as the NN.

The convention used to identify individual transients within a set is by designating the raw data file followed by each transient number (i.e. EXCA210:1,3-6 equates to file EXCA210, transients 1,3,4,5 and 6). The terms **X*i***, **X*i***$_t$ and **X*i*_*j*** (where *i*=1,2,3 and *j*=0,..,-5, -10, -15, -20), refer to the set of spectral coefficients composing the **X*i*** template, the **X*i*** template itself and the set **X*i*** degraded to have SNR levels which vary from 0 to -20 dB.

A. EXAMINATION OF TRANSIENT TYPES

1. The Reference Template

Time plots for all transients are shown in Figures 3.1 and 3.2. Note that each file of type EXCAXXX contains several transients of length 1020.

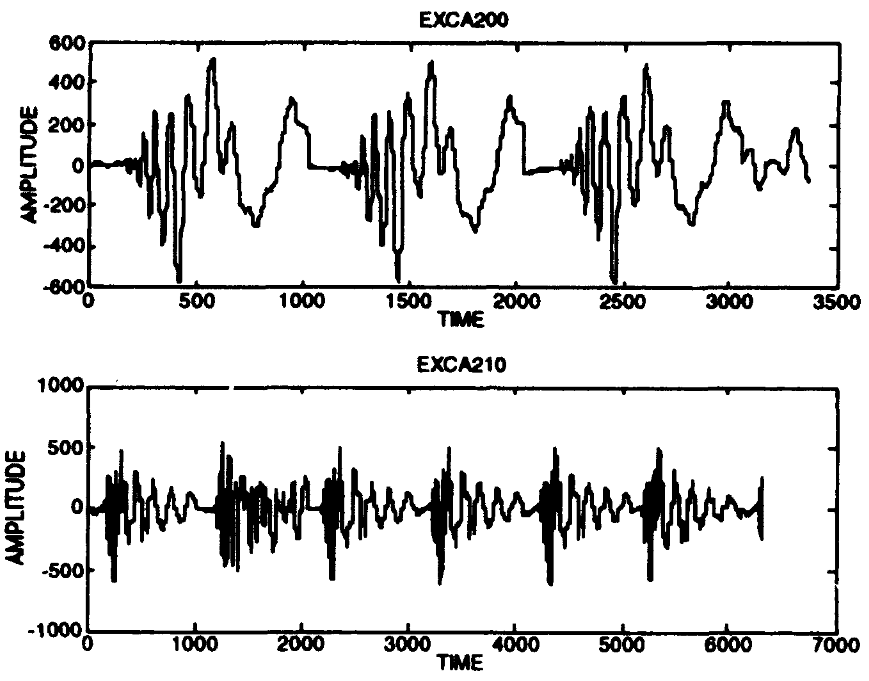


Figure 3.1: Time vs. Amplitude Plots for EXCA200 and EXCA210

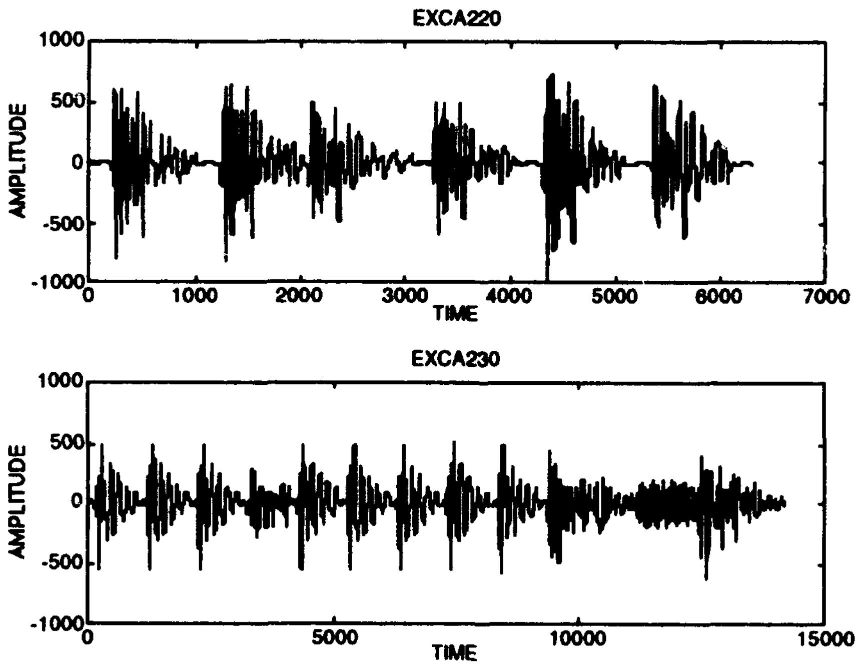


Figure 3.2: Time vs. Amplitude Plots for EXCA220 and EXCA230

The reference template is an averaged set of spectral coefficients produced by combining spectral coefficients obtained using similar transients from the data sets considered. Files EXCA210, 220 and 230 are sampled at a lower rate than EXCA200. In order to insure meaningful results, a uniform sampling rate is required. The sampling rate for EXCA210, 220 and 230 is slower by a factor of 2.5 compared to the rate of file EXCA200. Therefore to compensate for this change, the first 408 points are considered for transients within sets EXCA210, 220 and 230. After truncation to the first 408 samples, a "factor of five" interpolation and a "factor of two" decimation scheme are applied to bring the transient length back to 1020 points following the procedures given in [1]. This interpolation/decimation scheme allows the use of all transients given in the study.

The template construction involves an examination of time and frequency behavior for each transient. Figure 3.1 shows that set EXCA200 has three transients which have similar characteristics. When each raw data set is examined closely in the time domain, more similarities become obvious. Figure 3.3 shows the similarities between EXCA200:1 and EXCA230:1 over 1020 and 508 points respectively. EXCA220:1 however, is entirely different from EXCA200:1, EXCA210:1 and EXCA230:1 and is thus classified into a separate category. When all transients are examined within each raw data set, visual similarities become obvious and partial classifications follow.

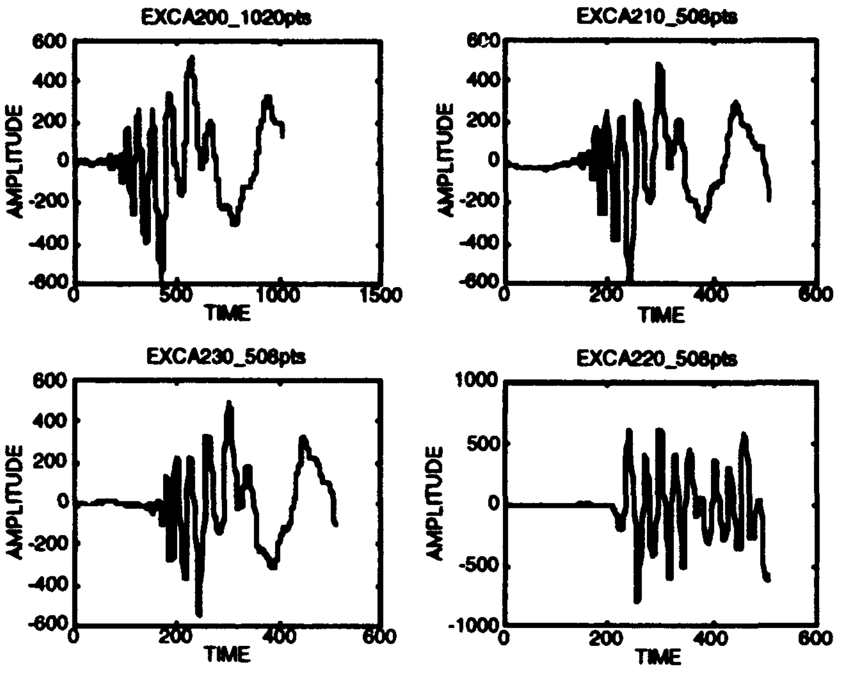


Figure 3.3 : Transient Classification in the Time Domain for EXCA200:1, EXCA210:1, EXCA230:1 and EXCA220:1

Classification is continued in the frequency domain to obtain spectral coefficients for further processing. Figures 3.4 and 3.5 show spectral similarities between transient sets EXCA210, EXCA220, EXCA200 and EXCA230. Spectral coefficient sets EXCA210:1,3-6, EXCA230:1-3,5-9 and EXCA200:1-3 are grouped into one file called **X1** shown in Figure 3.6. The average of file **X1** is used to compose the first template known as **X1**$_\mathbf{t}$. Spectral coefficients in set EXCA220:1-2,4-6 are similar and unique and are all grouped into one file called **X2** shown in Figure 3.7. Their resulting average is used to create the second template known as **X2**$_\mathbf{t}$. Finally, spectral coefficient sets EXCA210:2, EXCA220:3, EXCA230:4,10-13 are not similar to any other spectral coefficient sets and are grouped into set **X3**. The **X3** set is known as the "everything else" set

[1] and is shown in Figure 3.8. The resulting average is used to created the last template known as $X3_t$.

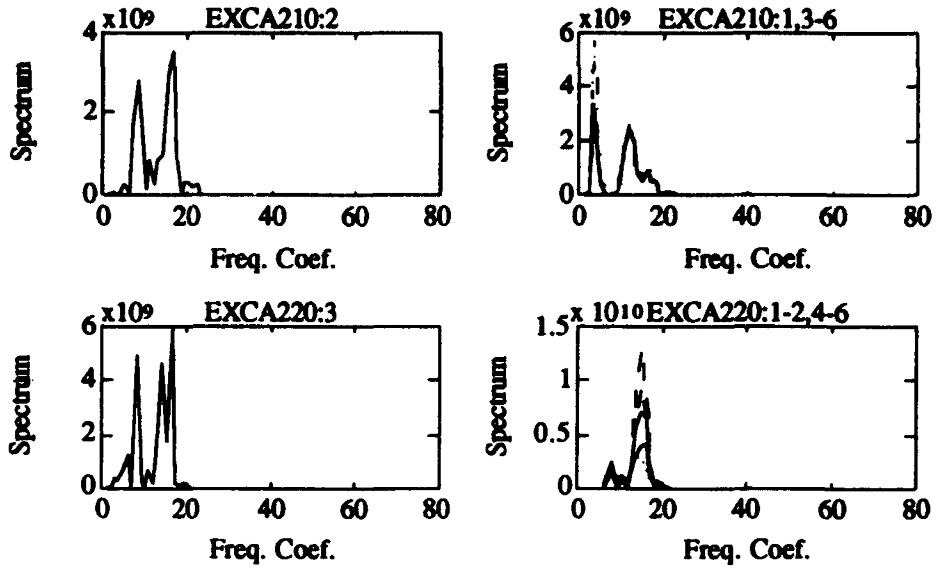


Figure 3.4: Spectra for Transients EXCA210 and EXCA220

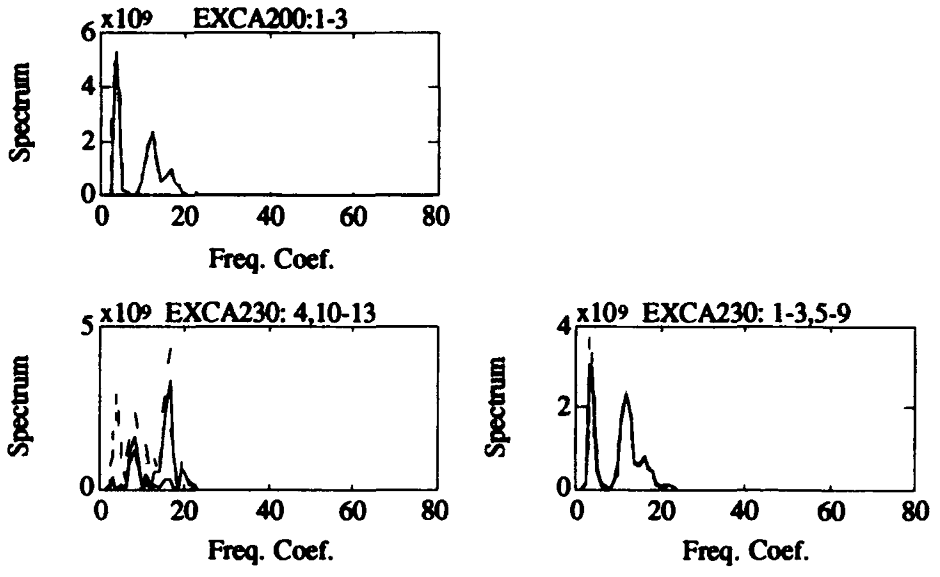


Figure 3.5: Spectra for Transients EXCA200 and EXCA230

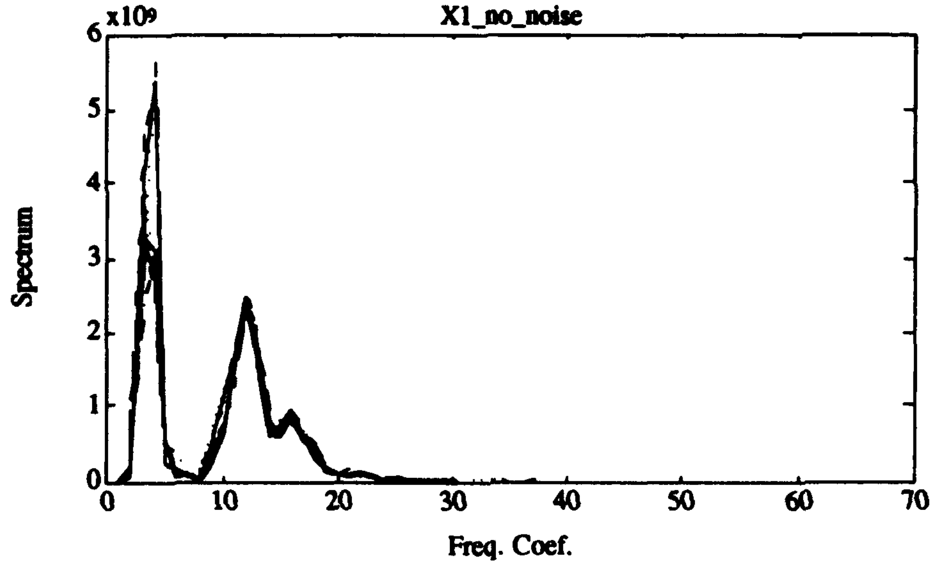


Figure 3.6: Superposition of All Spectral Coefficients Used in Template $X1_t$

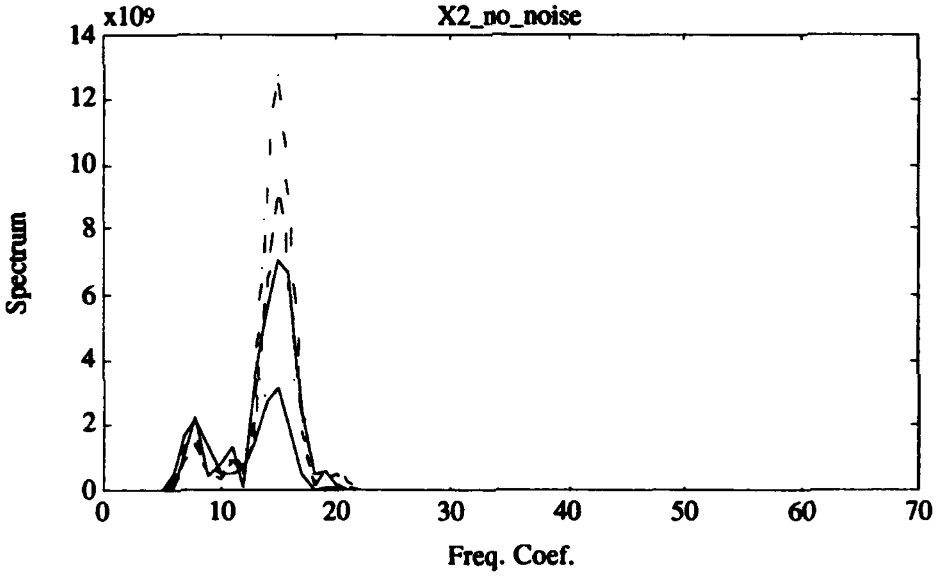


Figure 3.7: Superposition of All Spectral Coefficients Used in Template $X2_t$

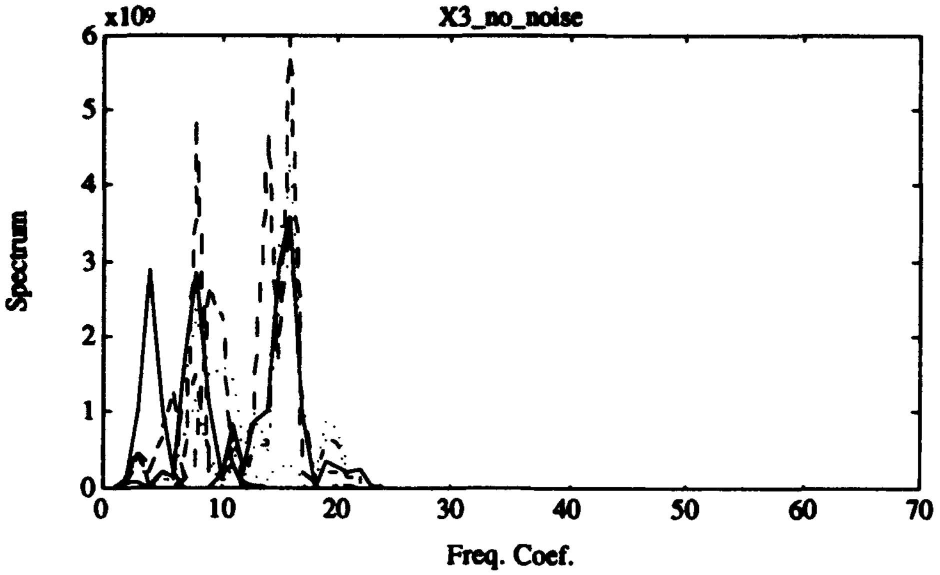


Figure 3.8: Superposition of All Spectral Coefficients Used in Template $X3_t$

Figures 3.9 and 3.10 are additional classification aid examples of information contained in the frequency domain. The upper left side of Figure 3.9 demonstrates superimposed spectra corresponding to each of the spectral coefficient sets in EXCA200. The upper right side of the Figure 3.9 shows contour plots of the same spectra. Note that all coefficient sets in the upper half of the Figure 3.9 are identical (a fact already apparent in the time domain plots). The lower half of Figures 3.9 and 3.10 display quite a different scenario. The bottom plots in Figure 3.9 represent frequency information for each transient in set EXCA210. The contour plot allowed us to determine that EXCA210:2 is different from all other transients in that signal. Observing the top half of Figure 3.9 for EXCA200 it becomes apparent that all spectral coefficient sets in transient set EXCA210 (with the exception of EXCA210:2) and all transient sets in EXCA200 are very similar. Reviewing the lower half of Figure 3.10 which represents transient set EXCA230, it becomes apparent that EXCA230:1-3,5-9 are similar to EXCA200:1-3. These spectral coefficient sets are grouped into file X1.

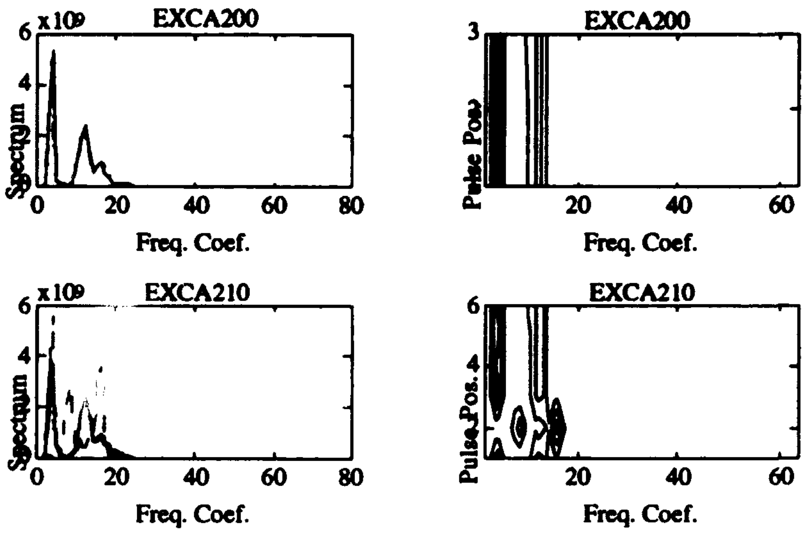


Figure 3.9: Spectral Information for sets EXCA200 and EXCA210

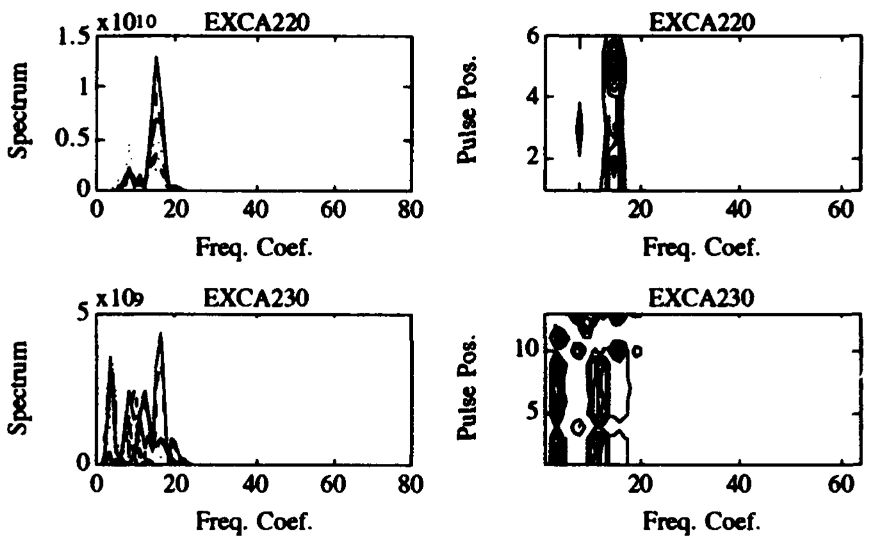


Figure 3.10: Spectral Information for sets EXCA220 and EXCA230

A similar process is used to determine the remaining groups. Transients EXCA220:1,2,4-6 constitute group **X2** as shown in Figure 3.7. Finally, after review of all remaining sets of spectral coefficients, the last group consists of EXCA210.2, EXCA220:3 and EXCA230:4,10-13 and is labeled **X3**, shown in Figure 3.8. Recalling that the information is also contained in the time domain plots, one can use either or both methods for classification.

2. Addition of Noise

The main thrust of this study is to determine the performance of the four spectral measure algorithms introduced earlier on a known set of signals with varying amounts of added white Gaussian noise. The resulting SNRs of the noisy transient considered are between -20 dB and 0 dB. Note that the added noise may not be the only noise present. The raw data sets may already contain small amounts of noise thereby effectively decreasing the SNR value to an even lower ratio. Therefore, the added noise to each set of spectral coefficients is measured conservatively as to having at least that amount of added noise and possibly more.

SNRs are produced by adding to individual **X*i*** sets a normally distributed random variable (zero mean, unit variance) multiplied by the power in each **X*i***, multiplied again by a specific dB scale factor corresponding to the SNR level desired. For example, an SNR level of -3 dB is achieved by multiplying a normally distributed random variable by the square root of the signal power and a scale factor of the square root of two. The desired noisy signal is obtained by using the equation :

$$\text{signal} + \text{noise} = \text{signal} + (\text{random\_number} * \text{scale\_factor} * \sqrt{\text{signal\_power}})\ . \qquad (1)$$

The scaling factors used in equation (1) are shown in Table 3.1 [5]. The signal flow diagram is represented in Figure 3.11 [5] [6].

dB	Scale Factors
0	1
- 1	1.123
- 2	1.265
- 3	1.414
- 4	1.581
- 5	1.789
- 6	2
- 7	2.236
- 8	2.509
- 9	2.828

TABLE 3.1: LOG / SCALE FACTOR CHART

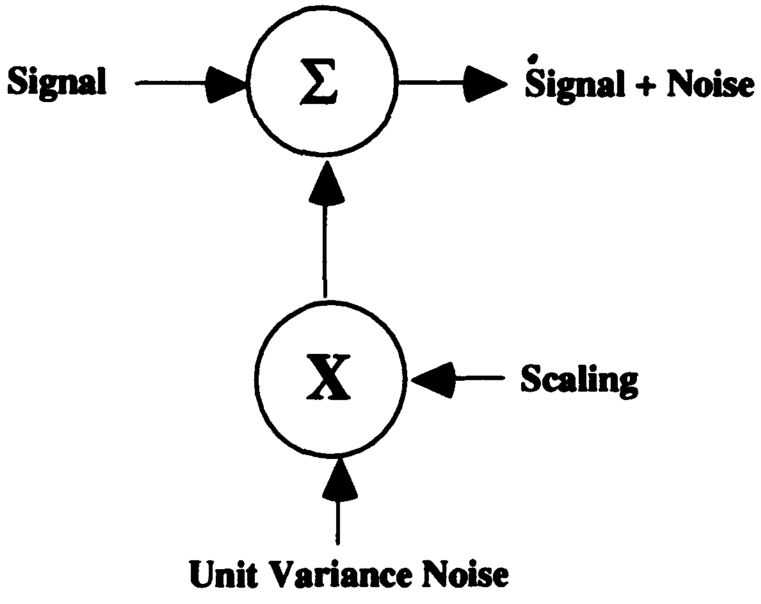


Figure 3.11: Signal Flow Diagram

Note that each transient within every set has noise added to it specifically based on its signal power. The signal power for each transient within a raw data set is determined by squaring each group of 1020 data points, summing the values then dividing by 1020 to determine the magnitude.

For a given transient, the power is computed by :

$$P_{signal} = \frac{1}{1020}\sum_{i=0}^{1019} x_i^2 \quad .$$

Figures 12 and 13 show the effects additive noise has on transient set EXCA200. Examples of SNR values of 0 dB through -5 dB, -10 dB and -15 dB of EXCA200 set are presented. Note, as the SNR decreases below 0 dB, the determination of the original transient set becomes visually impossible.

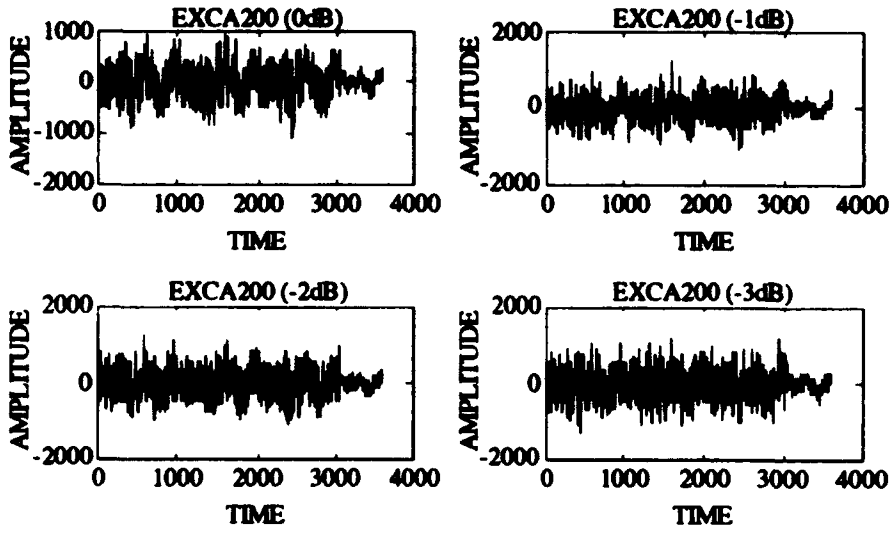


Figure 3.12: SNR Values of 0 dB Through -3 dB for EXCA200

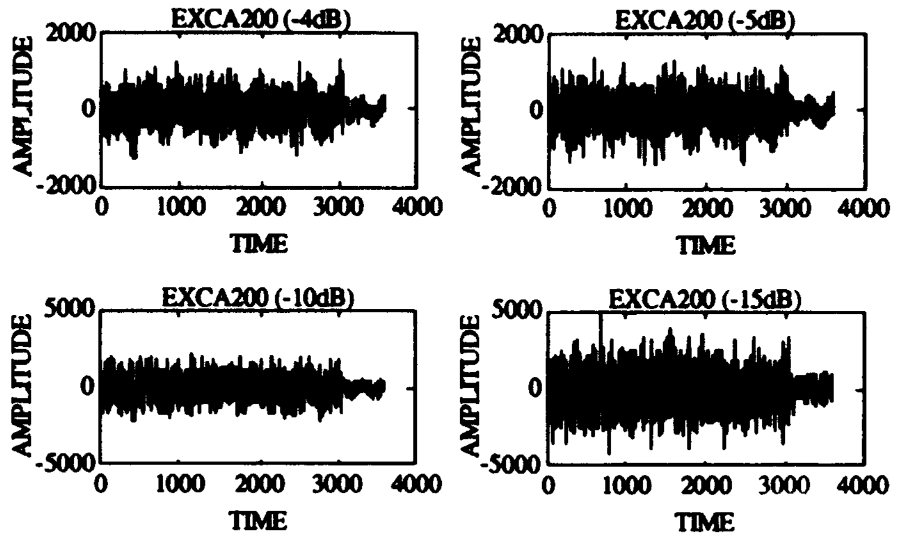


Figure 3.13: SNR Values of -4 dB Through -15 dB for EXCA200

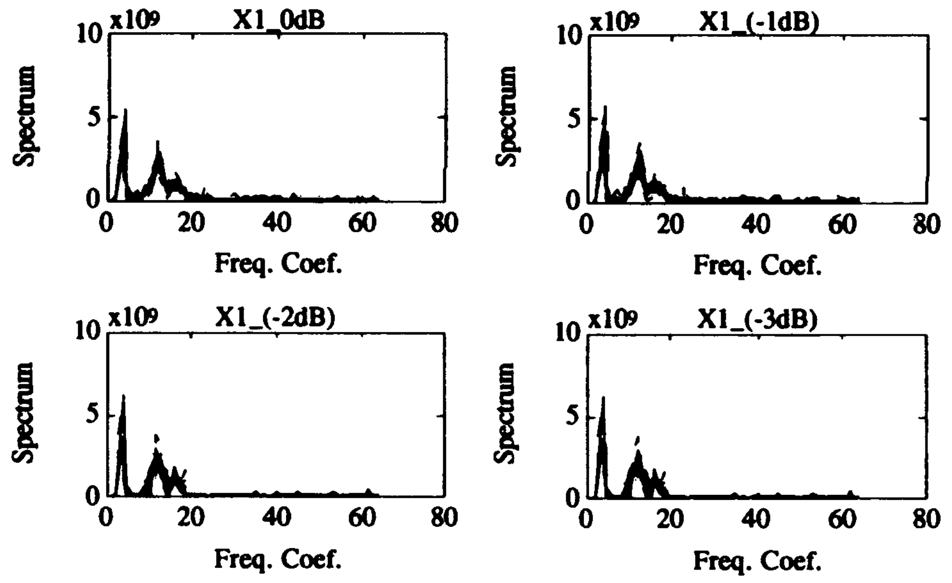


Figure 3.14: X1 with SNRs of 0 dB Through -3 dB

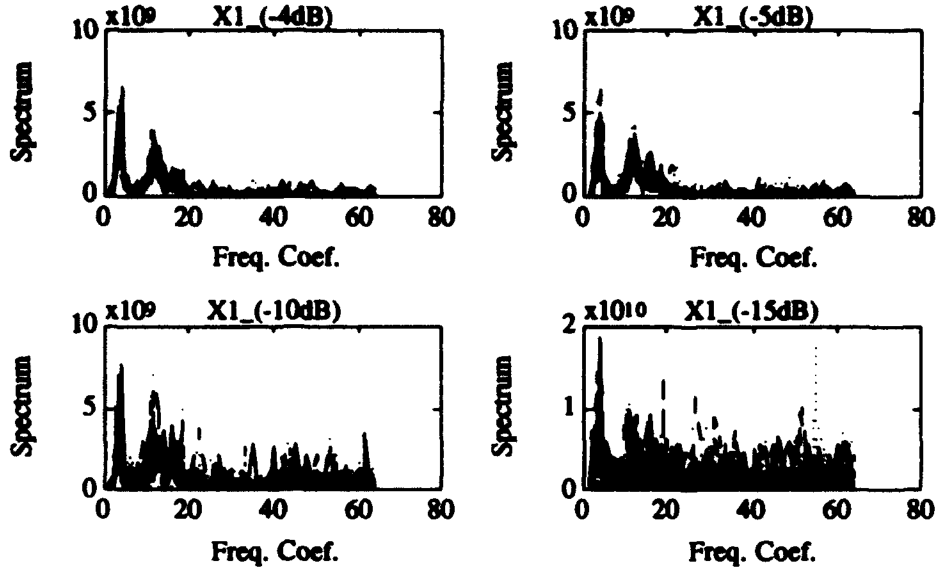


Figure 3.15: X1 with SNRs of -4, -5, -10, -15 dB

B. SPECTRAL-BASED STUDY

1. Noise-Free Analysis

The three spectral templates $X1_t$, $X2_t$ and $X3_t$ created in Part A of this chapter are correlated against their constituent members using all four spectral measure algorithms. The results are presented in Figures 3.16 through 3.22. The template and test file used for each correlation are designated in the bottom left-hand portion of each figure. The file on the left represents the template and the file on the right represents the test file. For example, in Figure 3.16 the template file used was X1_no_noise (or $X1_t$) and the test file used was X1_no_noise (or X1).

Only the first 64 magnitude squared Fourier coefficients are used in this investigation as these spectral coefficients contain most of the information. There are four designations for the spectral measures used in each figure. The first designation is "cross-correlation 1 - ur ", which represents the results obtained using the normalized cross-correlation spectral-based measure. The second designation is "cross-correlation 2 - rr ", which represents the results using the pre-processed data sequence formed from using the spectral coefficients. Note, the mean of the sequence first formed by the spectral coefficients is removed before the normalized cross-correlation values are computed. The third and fourth spectral measure techniques used are the Bhattacharyya distance and the divergence. They are represented in the bottom left-hand and right-hand portions of each figure, respectively.

Figures 3.16 and 3.17 represent the first set of correlation results. As expected, the spectral measures for files **X1$_t$** and **X2$_t$,** when correlated with **X1** and **X2** respectively, have high values (i.e. 0.9 to 1.0). The Bhattacharyya distance, with a maximum distance value of 1, was also high for the same test files. The divergence plots with possible values ranging from 0.0 to a maximum of 20 (in some cases), show results in the range of 0.05 to 0.3 with a mean of approximately 0.12.

The **X3$_t$** compared with **X3** had somewhat different results. Recall from Part A in this chapter that **X3** was primarily composed of spectral coefficients that did not fit into either **X1** or **X2** sets. In essence, **X3** is filled with "all the rest". Figure 3.19 shows the first three spectral measures range from 0.2 to 0.98, the divergence ranges from 0.1 to 5.0. Note, because **X3$_t$** and **X3** do not correlate consistently with each other, **X3$_t$** can not be applied as a usable template. However, for purposes of completeness and comparison, this study includes **X3$_t$** and **X3** in all results.

The remaining figures in this section represent comparisons of each template with a different set. Figures 3.19 and 3.20 represent the results obtained

by correlating $X1_t$ with X2 and $X1_t$ with X3 respectively. As expected, results show a decrease in spectral measure values from results obtained in Figures 3.16 and 3.17. The cross-correlation (both rr and ur) and Bhattacharyya distance measures are much smaller and the divergence measures are much larger. Figures 3.21 and 3.22 represent a similar trend in the results for the correlation of $X2_t$ with X1 and $X2_t$ with X3 respectively. Poor correlation results are anticipated because the frequency information contained in each file should not correlate.

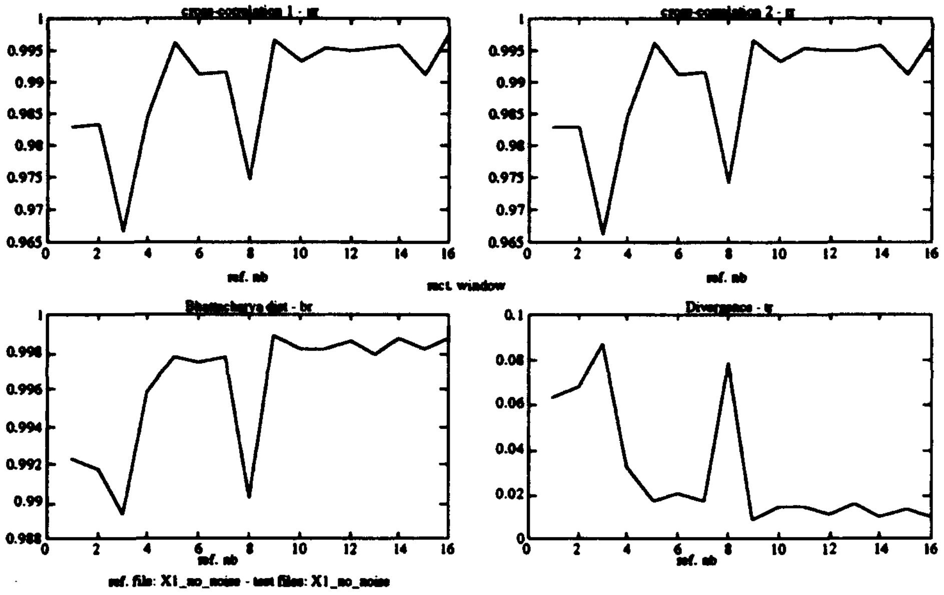


Figure 3.16: Spectral Measure Results for $X1_t$ Versus X1

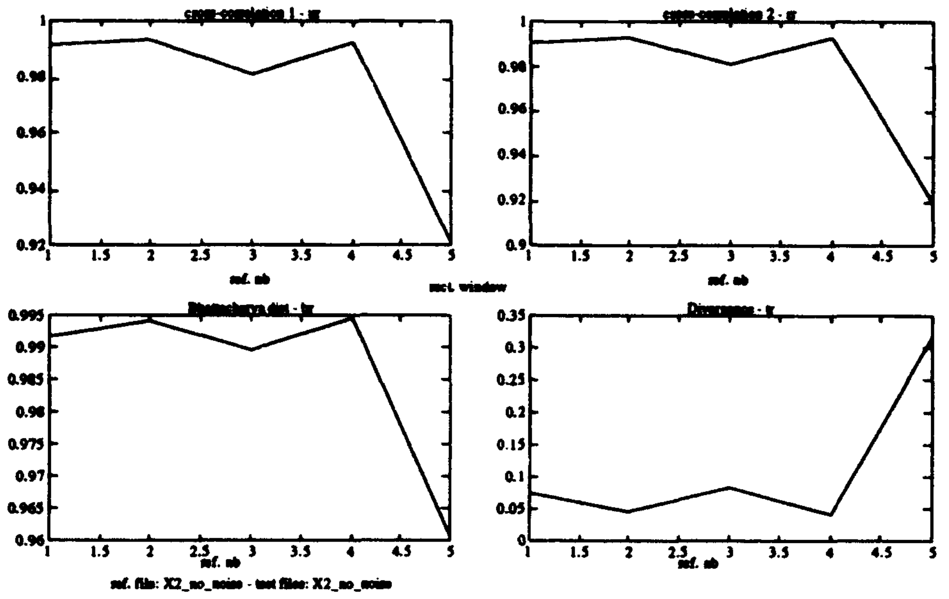


Figure 3.17: Spectral Measure Results for $X2_t$ Versus X2

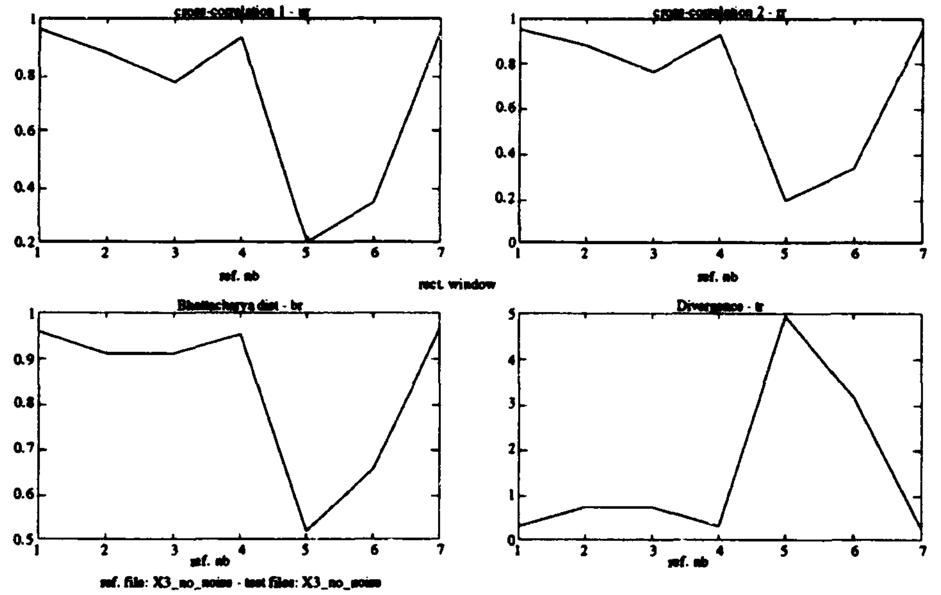


Figure 3.18: Spectral Measure Results for $X3_t$ Versus X3

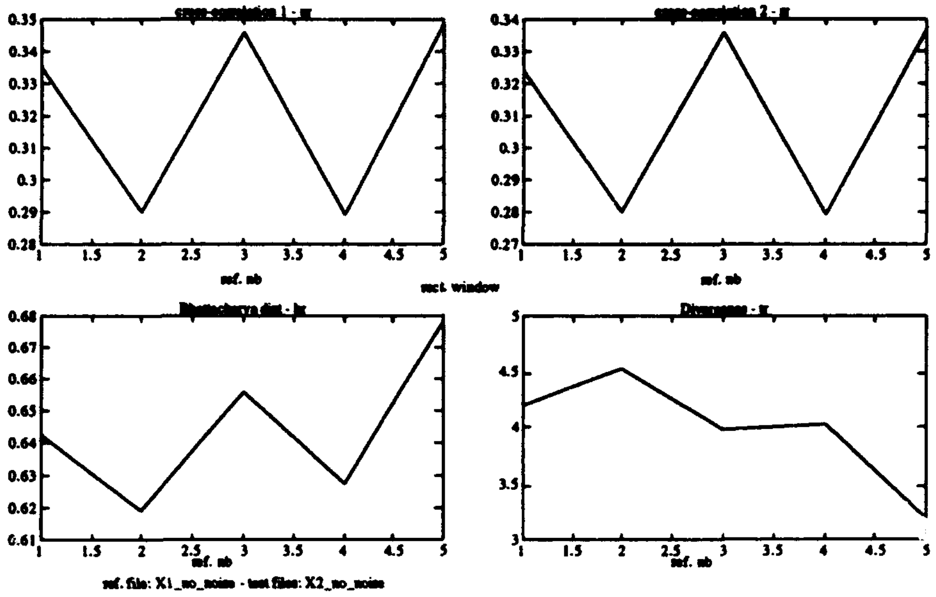


Figure 3.19: Spectral Measure Results for $X1_t$ Versus X2

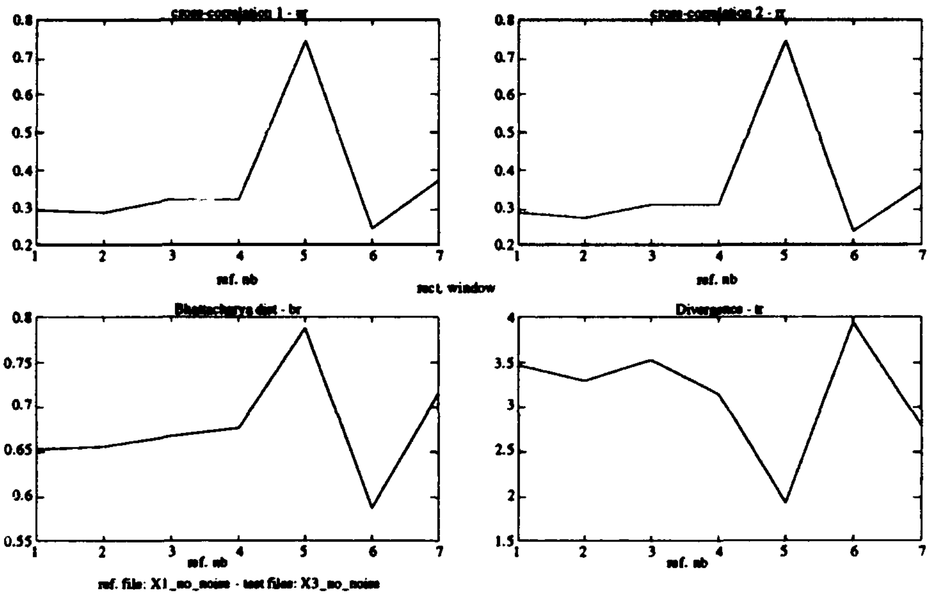


Figure 3.20: Spectral Measure Results for $X1_t$ Versus X3

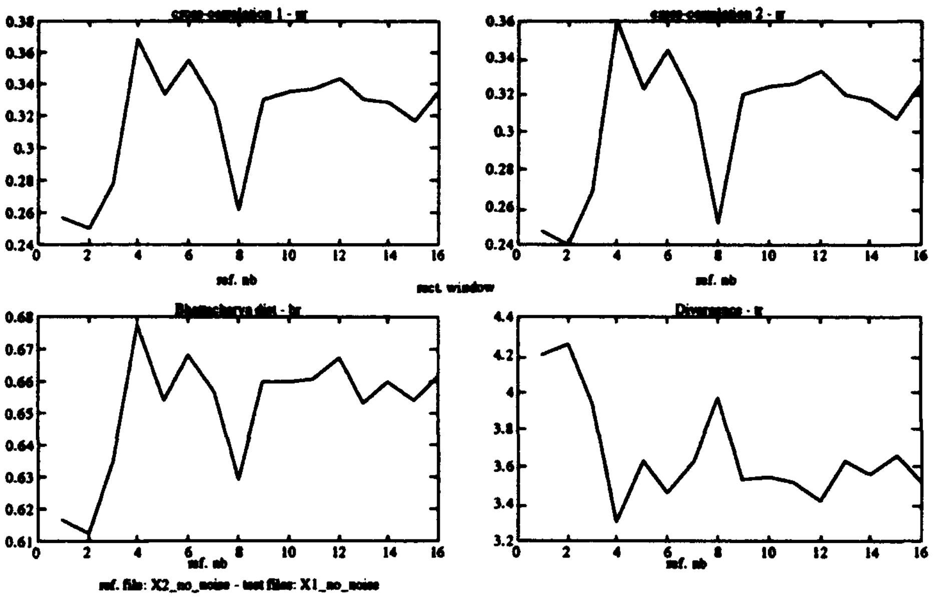


Figure 3.21: Spectral Measure Results for X2$_t$ Versus X1

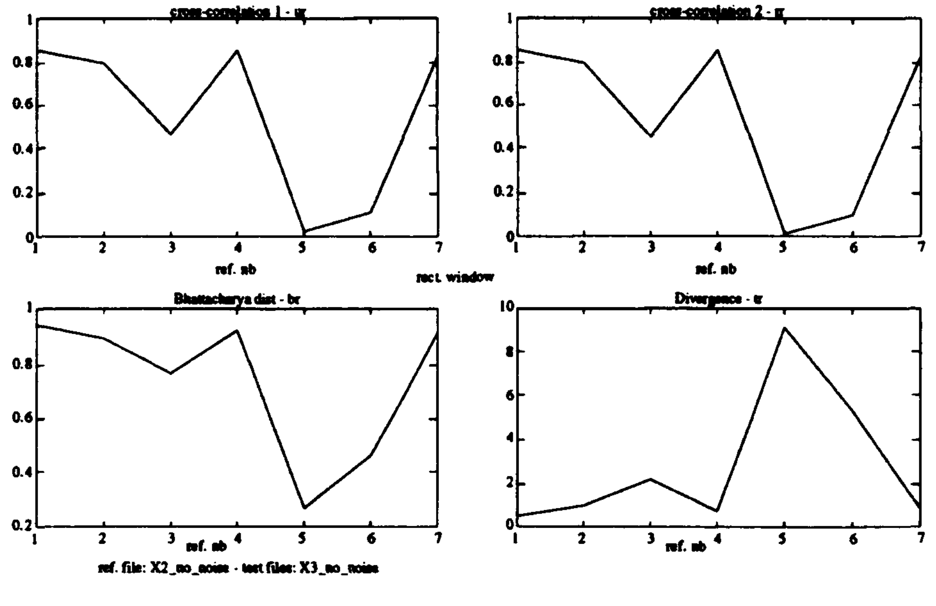


Figure 3.22: Spectral Measure Results for X2$_t$ Versus X3

The above comparisons reveal that information regarding the degree of similarity between transients can be obtained from all four spectral measures. Both cross-correlation-1-ur and cross-correlation-2-rr each have straightforward algorithms and are easy to compute and interpret. The values range between [0,1] and [-1,1] respectively. Special attention is placed on cross-correlation-2-rr technique. The values obtained via this measure are typically the same as the cross-correlation-1-ur for files constructed of the same sets of spectral coefficients and typically lower for files constructed of different sets of spectral coefficients. Throughout this study, all four spectral measures will be used. However, cross-correlation-2-rr appears to be the more robust technique.

2. Signal Plus Noise Analysis

a. Single Run

As mentioned in Part A.2 of this chapter, the main thrust of this study is to measure the performance of all four spectral measure algorithms on a known set of signals with varying amount of added white Gaussian noise. In this section, SNRs from 0 dB to -5 dB, -10 dB, -15 dB and -20 dB are obtained for individual transients within each raw set. The resultant "noisy signal" is processed in the same fashion as before to obtain spectral coefficient groups.

Summary plots are presented in Figure 3.23 through Figure 3.34. Each plot represents the average spectral measure for sets with SNR values of 0 dB to -5 dB, -10 dB, -15 dB and -20 dB. The horizontal lines represent the average value of the original spectral measure. The reference files for each plot are labeled **X*i*–X*j*#db**. This indicates that the *i*th template is used against the *j*th data file and is indexed by the SNR which can be read off the horizontal axes. The upper left-hand portion of each figure shows spectral measure results for varying SNRs of three individual comparisons. The remaining three graphs in each figure show individual plots for comparison. A vertical line is drawn between the minimum and maximum values obtained for each SNR during the averaging calculations, and is superimposed over the respective SNR value.

For example, Figure 3.23 displays the average results using the cross correlation spectral measure technique for $\mathbf{X1_t}$ correlated with **X1_*j*** (i.e. X1_X1#db, represented by the solid line), $\mathbf{X2_t}$ correlated with **X2_*j*** (i.e. X2_X2#db, represented by the dashed line) and $\mathbf{X3_t}$ correlated with **X3_*j*** (i.e. X3_X3#db, represented by the dotted line). The upper left-hand corner of the first graph shows which files are used in the correlation. The remaining three graphs labeled X1 band, X2 band and X3 band, representing the $\mathbf{X1_t}$, $\mathbf{X2_t}$ and $\mathbf{X3_t}$ respectively, show individual average correlation with the addition of minimum and maximum value lines. Results show that spectral measure values tend to decrease rapidly at around -5 dB for the $\mathbf{X1_t}$, $\mathbf{X2_t}$ correlations and around -3 dB for the $\mathbf{X3_t}$ correlation. One should pay close attention to the minimum and maximum values in the remaining three plots in each figure as they indicate the amount of swing to the average curve at each SNR point. In particular, the $\mathbf{X1_t}$, and $\mathbf{X2_t}$ spectral measure curves located in the upper right-hand corner and lower left-hand corner of Figure 3.23 respectively, show small differences between the minimum and maximum values from SNRs of 0 dB to -5 dB indicating that the average plot in the first graph between SNRs of 0 dB and-5 dB is very close to a true spectral measure plot and thus very accurate. Whereas for $\mathbf{X3_t}$, comparisons in the last plot (lower right-hand corner) of Figure 3.23, show wide variations for the same SNR values, indicating that the true spectral measure for that SNR range is not accurately delineated in the average plot representation. Recall that **X3** is composed of "all-the-rest" transients from the raw data sets, thus giving poor results.

See Appendix A for each $\mathbf{X}\boldsymbol{i}_\mathbf{t}$ correlation with each **Xi_j** set. These lead to the summary plots presented in Figures 3.23 through 3.34.

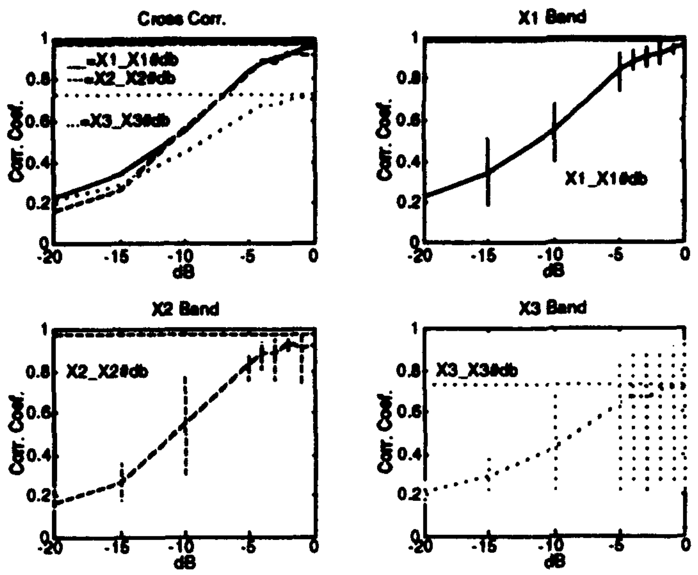


Figure 3.23: Cross-Correlation Measure Results for X1$_t$, X2$_t$ and X3$_t$ as Compared to X1_*j*, X2_*j*, and X3_*j* Respectively, Averaged for 1 Run

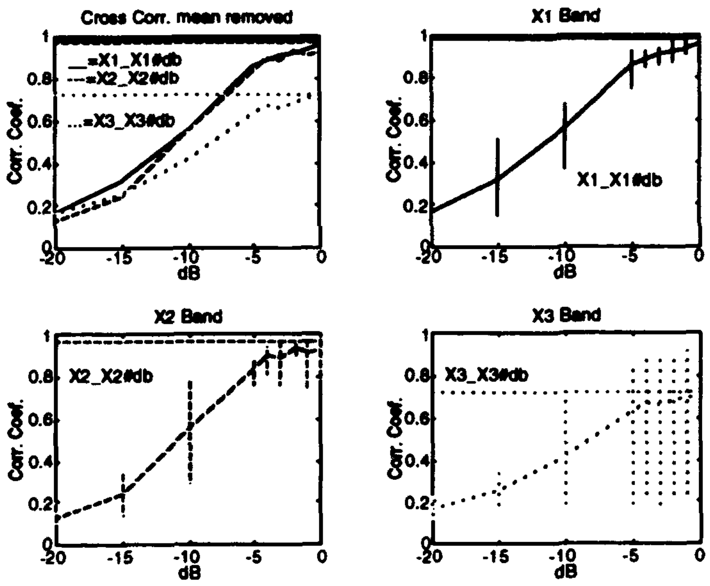


Figure 3.24: Cross-Correlation (mean removed) Measure Results for X1$_t$, X2$_t$ and X3$_t$ as Compared to X1_*j*, X2_*j*, and X3_*j* Respectively, Averaged for 1 Run

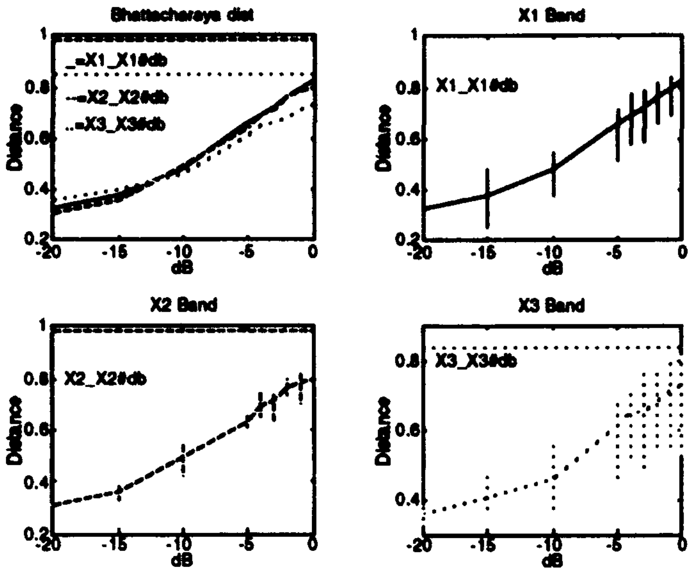


Figure 3.25: Bhattacharyya Measure Results for X1$_t$, X2$_t$ and X3$_t$ as Compared to X1_*j*, X2_*j*, and X3_*j* Respectively Averaged for 1 Run

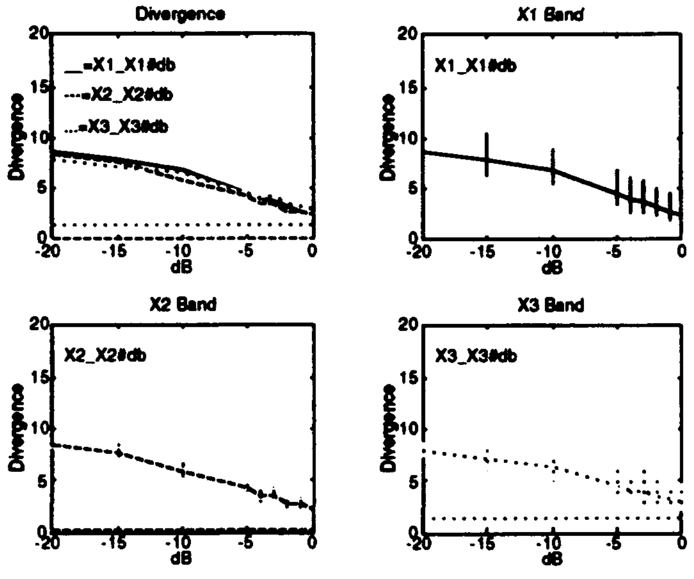


Figure 3.26: Divergence Measure Results for X1$_t$, X2$_t$ and X3$_t$ as Compared to X1_*j*, X2_*j*, and X3_*j* Respectively, Averaged for 1 Run

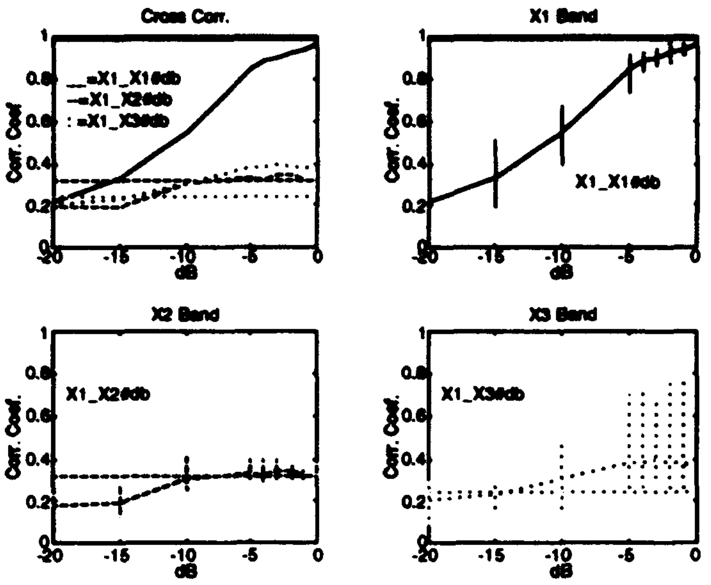


Figure 3.27: Cross-Correlation Measure Results for $X1_t$ as Compared to X1_j, X2_j, and X3_j Respectively, Averaged for 1 Run

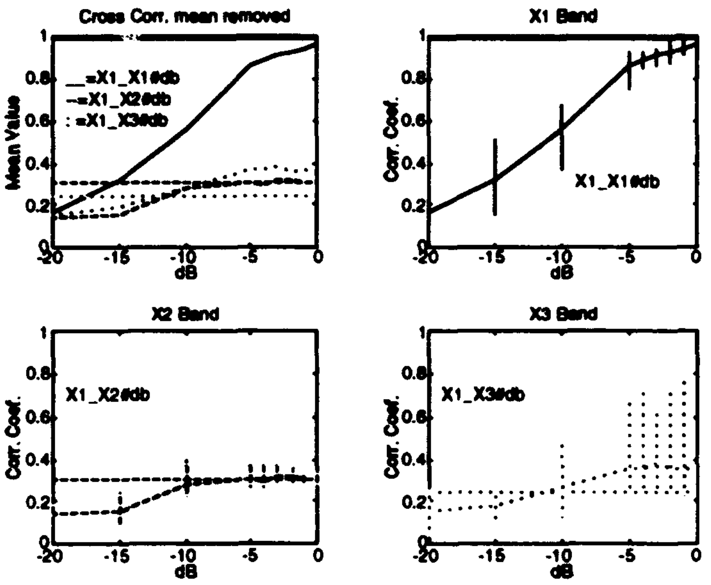


Figure 3.28: Cross-Correlation (mean removed) Measure Results for $X1_t$ as Compared to X1_j, X2_j, and X3_j Respectively, Averaged for 1 Run

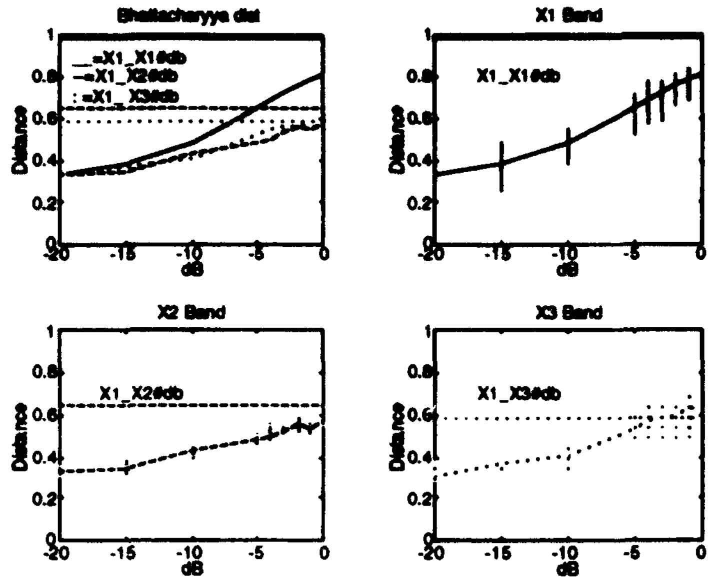


Figure 3.29: Bhattacharyya Measure Results for X1t as Compared to X1_j, X2_j, and X3_j Respectively, Averaged for 1 Run

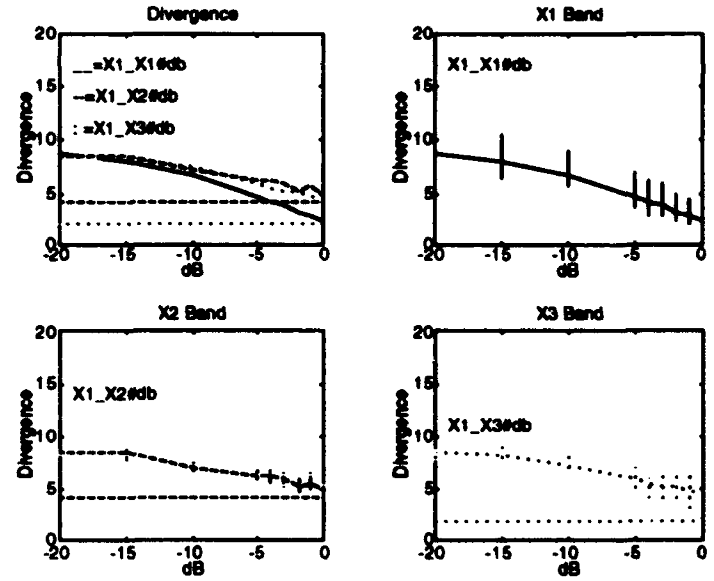


Figure 3.30: Divergence Measure Results for X1t as Compared to X1_j, X2_j, and X3_j Respectively Averaged for 1 Run

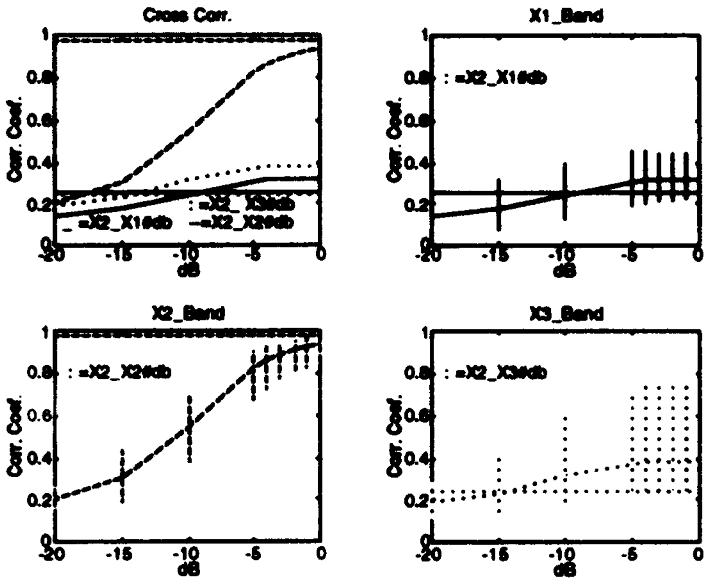


Figure 3.31: Cross-Correlation Measure Results for $X2_t$ as Compared to X1_*j*, X2_*j*, and X3_*j* Respectively Averaged for 1 Run

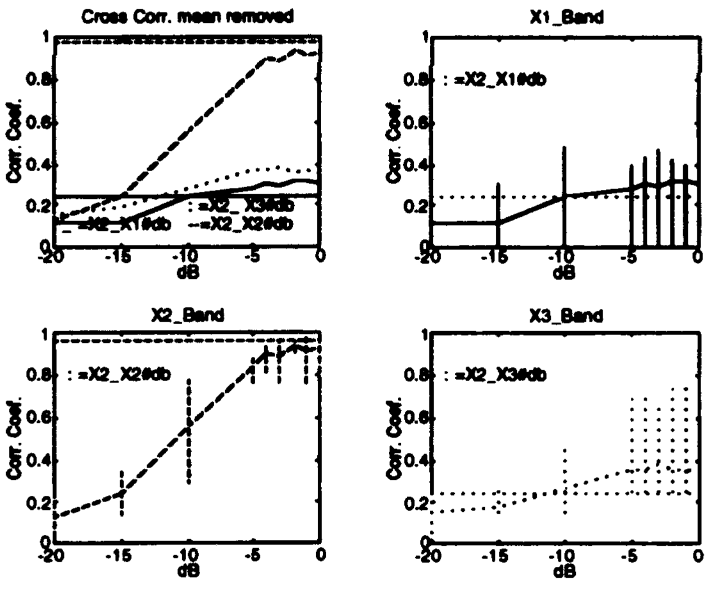


Figure 3.32: Cross-Correlation (mean removed) Measure Results for $X2_t$ as Compared to X1_*j*, X2_*j*, and X3_*j* Respectively Averaged for 1 Run

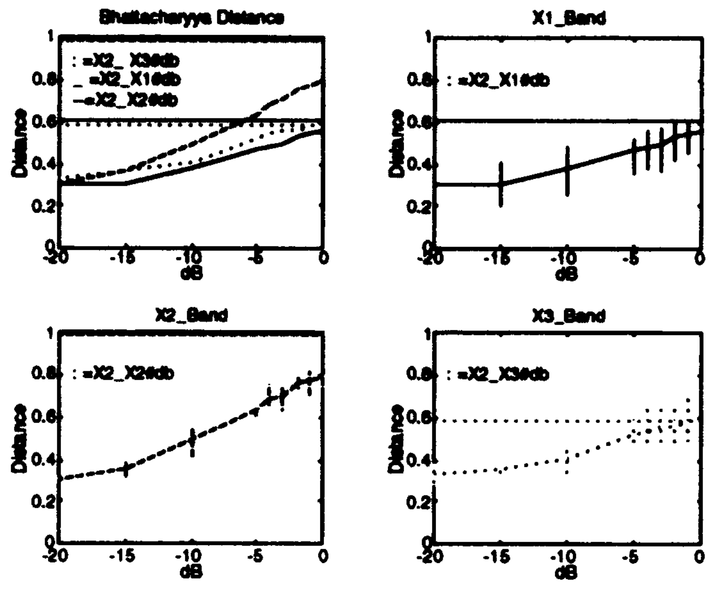


Figure 3.33: Bhattacharyya Measure Results for $X2_t$ as Compared to X1_*j*, X2_*j*, and X3_*j* Respectively Averaged for 1 Run

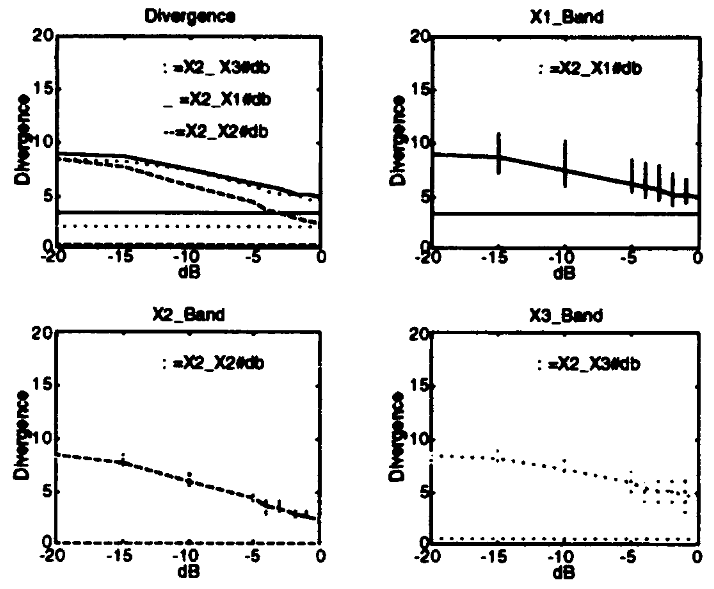


Figure 3.34: Divergence Measure Results for $X1_t$, $X2_t$ and $X3_t$ as Compared to X1_*j*, X2_*j*, and X3_*j* Respectively Averaged for 1 Run

The summary graphs in Figures 3.23 through 3.34 provide a way of viewing degradation of each spectral measure technique under different noise conditions. Results show in Figures 3.23 and 3.24 that as SNR decreases from 0 dB to -5 dB both cross-correlation measures retain their signal recognition integrity. An approximate correlation coefficient value of 0.8 was obtained for $X1_t$ compared with X1_5 and $X2_t$ compared with X2_5 (Figure 3.23). The normalized cross-correlation technique with the mean removed displays a slightly better performance at values of -15 dB and -20 dB SNR. Overall, this spectral measure exhibits the most robust results.

The Bhattacharyya distance and divergence do not have the degree of performance as the cross-correlation techniques. Both spectral measure techniques have poor performance and are difficult to use. For example, Figures 3.25 and 3.26 demonstrate the difficulty in interpreting results by showing a 0 dB value for the Bhattacharyya distance at approximately 0.82 decreasing to 0.67 at -5 dB, whereas the divergence starts at approximately 2.5 at 0 dB and increases to 5 at -5 dB.

Both cross-correlation techniques retain their integrity when comparing templates with other noisy signals of different composition. In order for a correlation technique to be valid, signals of one composition should show low correlation values when compared to a set of signals with different compositions. Indeed, the results in Figures 3.27 through 3.30 show that when $X1_t$ is compared with X2_*j* and X3_*j*, cross-correlation values range from approximately 0.39 at 0 dB in Figure 3.27 down to 0.18 at -20 dB in Figure 3.28. Bhattacharyya distance measures are comparatively small whereas the divergence values are high. Similar results are obtained for $X2_t$ as compared to X1_*j* and X3_*j*, in Figures 3.31 through 3.34.

b. Multiple Runs

In order to obtain statistically more reliable results, multiple consecutive simulations are completed. Each time the MATLAB® random number generator is accessed, a different value is obtained. The random numbers are used in the noise generation algorithm to obtain many different SNR realizations. Thus a large number of possible outputs are generated allowing the computation of the desired statistics. This is accomplished by automating all of the algorithms to loop and store all successive run values. After a specified number of simulations is completed, all stored values are averaged and displayed in Figures 3.35-3.46.

Results after 100 runs tend to smooth out any jagged edges of the curves produced in the single run case. The normalized cross-correlation coefficient (mean removed) remains the most robust member of the four spectral measures.

Figures 3.35 *through* 3.46 display the summary results for 100 consecutive runs. Note that all values displayed are averaged values. The vertical lines represent minimum and maximum values. These lines are now averages of the minimum and maximum values calculated for each spectral measure within each iteration.

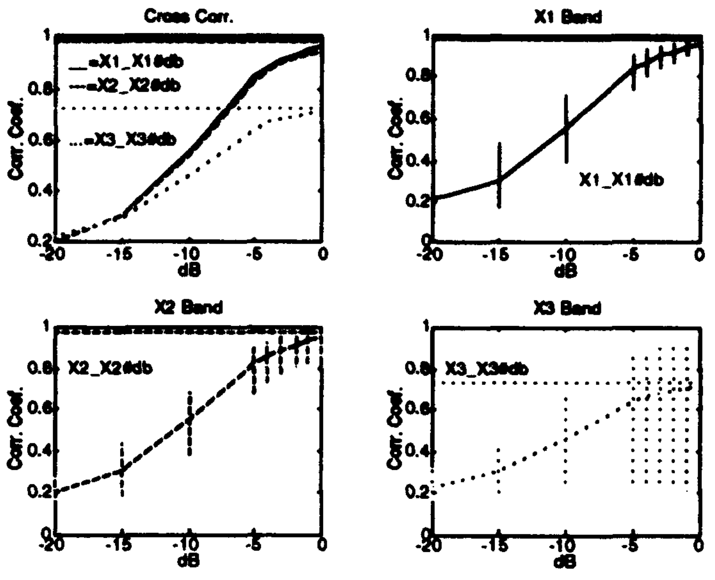


Figure 3.35: Cross-Correlation Results for X1$_t$, X2$_t$ and X3$_t$ versus X1_*j*, X2_*j*, and X3_*j* Respectively (Averaged for 100 Runs).

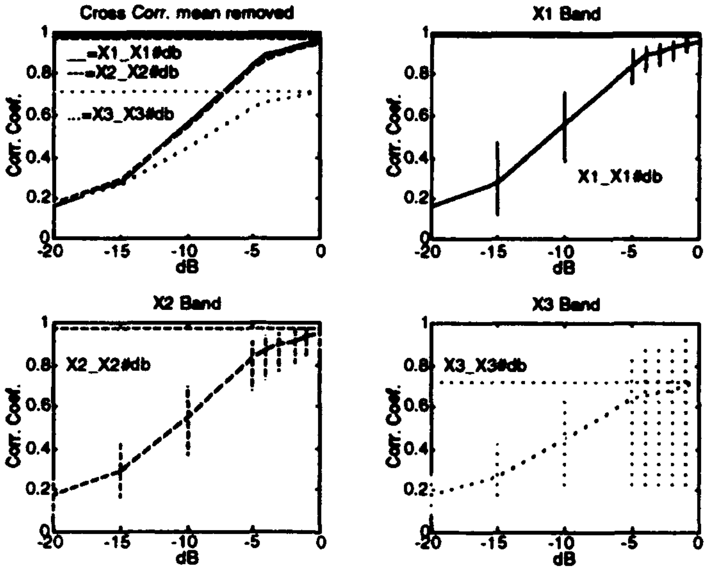


Figure 3.36: Cross-Correlation (mean removed) Measure Results for X1$_t$, X2$_t$ and X3$_t$ versus X1_*j*, X2_*j*, and X3_*j* Respectively (Averaged for 100 Runs).

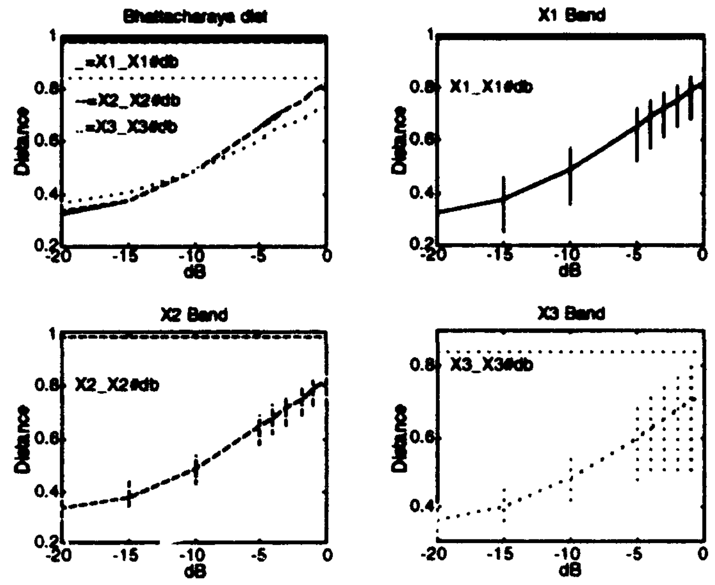


Figure 3.37: Bhattacharyya Results for X1$_t$, X2$_t$ and X3$_t$ versus X1_*j*, X2_*j*, and X3_*j* Respectively (Averaged for 100 Runs).

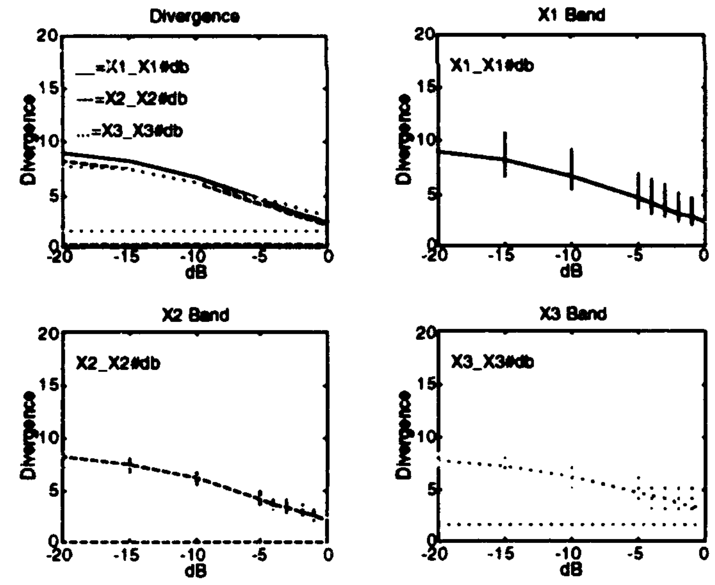


Figure 3.38: Divergence Results for X1$_t$, X2$_t$ and X3$_t$ versus X1_*j*, X2_*j*, and X3_*j* Respectively (Averaged for 100 Runs).

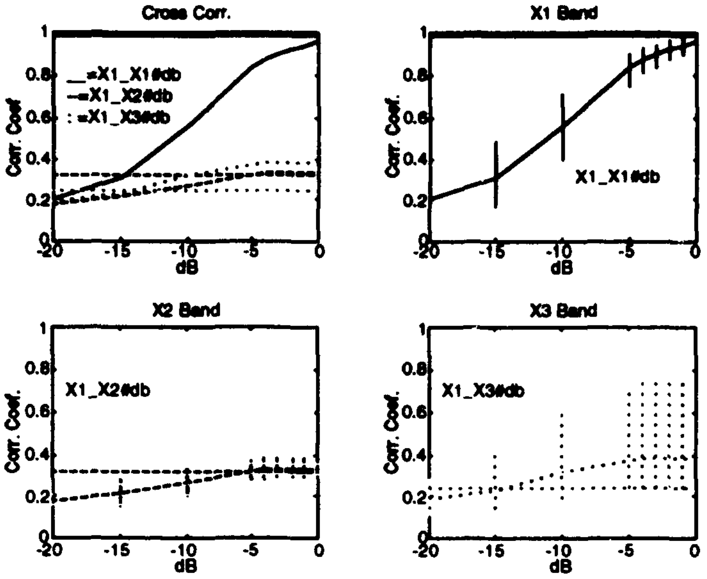


Figure 3.39: Cross-Correlation Results for X1t versus X1_*j*, X2_*j*, and X3_*j* Respectively (Averaged for 100 Runs).

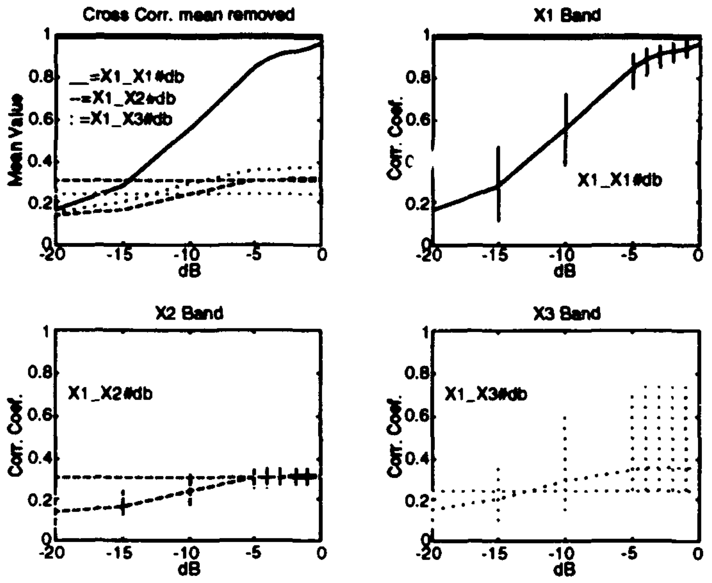


Figure 3.40: Cross-Correlation (mean removed) Results for X1t versus X1_*j*, X2_*j*, and X3_*j* Respectively (Averaged for 100 Runs).

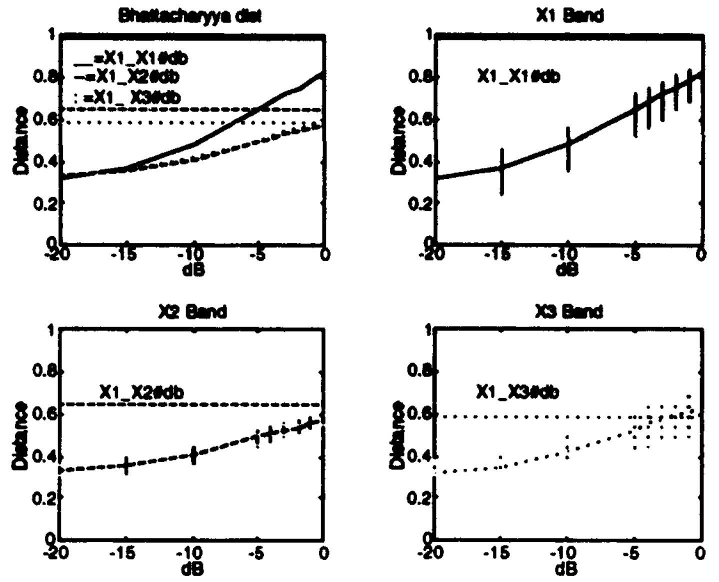


Figure 3.41: Bhattacharyya Results for X1$_t$ versus X1_*j*, X2_*j*, and X3_*j* Respectively (Averaged for 100 Runs).

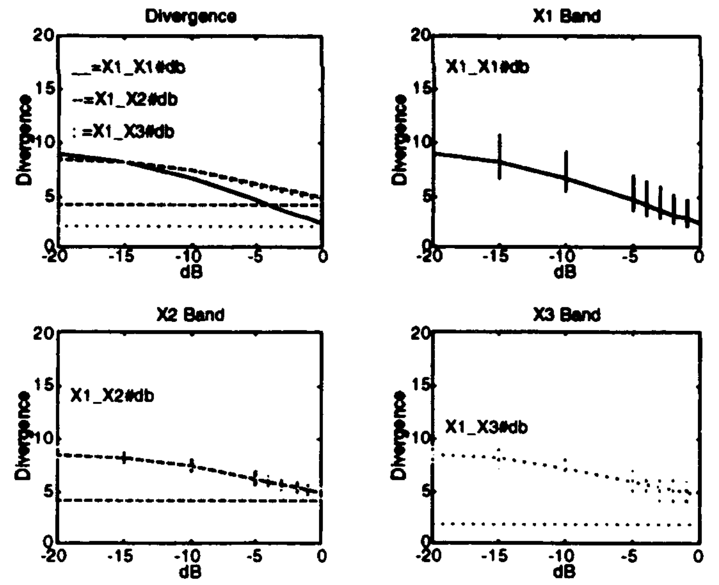


Figure 3.42: Divergence Results for X1$_t$ versus X1_*j*, X2_*j*, and X3_*j* Respectively (Averaged for 100 Runs).

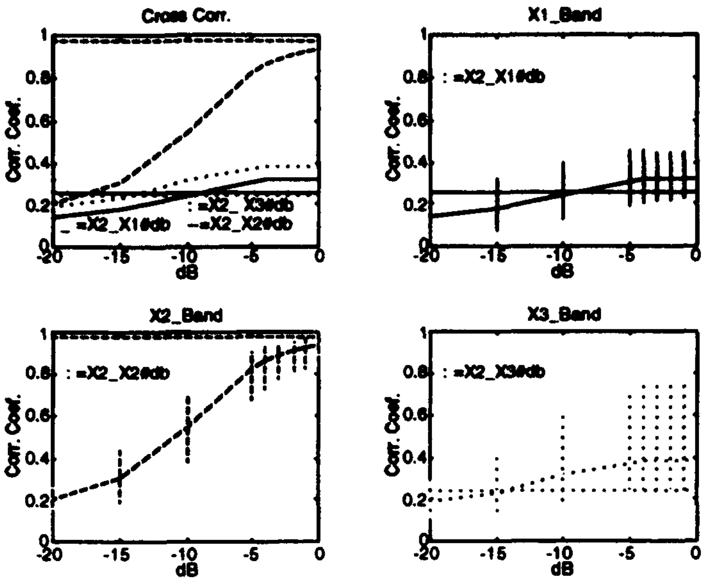


Figure 3.43: Cross-Correlation Results for X2t versus X1_j, X2_j, and X3_j Respectively (Averaged for 100 Runs).

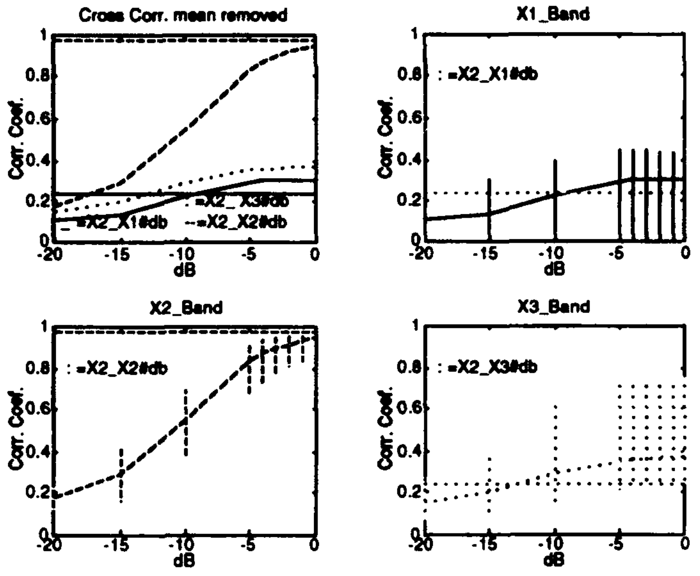


Figure 3.44: Cross-Correlation (mean removed) Results for X2t versus X1_j, X2_j, and X3_j Respectively (Averaged for 100 Runs).

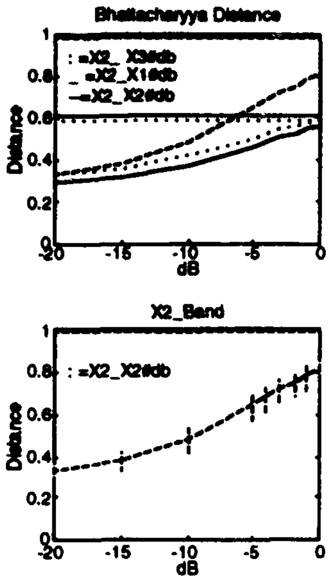


Figure 3.45: Bhattacharyya Results for X2$_t$ versus X1_*j*, X2_*j*, and X3_*j* Respectively (Averaged for 100 Runs).

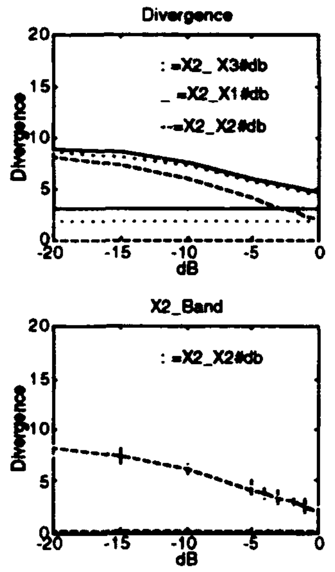


Figure 3.46: Divergence Results for X1$_t$, X2$_t$ and X3$_t$ versus X1_*j*, X2_*j*, and X3_*j* Respectively (Averaged for 100 Runs).

IV. NEURAL NETWORK STUDY

The purpose of this section is to investigate the potential applications of NN techniques to the automatic classification of the different transient types under study. NNs are particularly useful in circumstances where knowledge of a system or signal under test is not complete [1]. A NN can be "trained" when given a specific set of inputs with known outputs. The system will then "learn" to recognize all input training values and give a specified output indicating the degree to which it has learned. One potential disadvantage of the NN is the amount of time taken to adequately train a new system. In some complicated cases training could take hours depending upon processor architecture and clock speed. In this study, training was done operating on an Intel 486 processor with a clock speed of 66 MHz using 13 sets of spectral measure coefficients over 50,000 cycles, lasting a total of 45 minutes.

The spectral coefficients computed in Chapter III are used to train the NN to distinguish between X1_*j*, X2_*j* and X3_*j*. The back-propagation software package *NeuralWorks Professional II / Plus* ™ from NeuralWare, Inc. is used for this implementation. One hidden layer with ten processing elements (PE), delta rule and sigmoid transfer function with a learning momentum of 0.4 are used in the network configuration. The number of inputs are limited to the first 30 spectral coefficients. Figures 3.6, 3.7 and 3.8 show that most of the energy is concentrated in the first 30 coefficients, hence little information is lost with this restriction.

A. NEURAL NETWORK IMPLEMENTATION

1. Single Run

Training of the network is completed using the first eight transients within X1 and all five transients within X2. Because of its diverse transient composition, no transients from X3 are used for testing at this time. Tables 4.1a and 4.1b represent actual training set data presented to the network. Two training set types are used to determine which produce the best results. The first training method uses linear spectral coefficients. The second training method reduces the size of each spectral coefficient by taking $10*\log_{10}$ of each coefficient. This provides the NN a set of numbers that have a smaller dynamic range. Smaller coefficients allow the NN to classify each training set with more accuracy.

Results show that the second method of training the NN produces the best results. The first group of eight sets of numbers in Tables 4.1a and 4.1b are the first 30 spectral coefficients expressed in dB from each of the first eight transients within the $X1_t$. The second set of numbers in Table 4.1b represent the first 30 spectral coefficients expressed in dB from all five transients in the $X2_t$. In total, 13 transients are used to train the NN.

NEURAL NETWORK TRAINING FILE

(X1 pulses 1-6)

```
-0.10339750 0.03634666 0.04457024 0.04512217 0.03891442 0.03814090
&  0.03696092 0.03236073 0.04016165 0.04203824 0.04281557 0.04350580
&  0.04276869 0.04072964 0.04112998 0.04162601 0.04071784 0.03955620
&  0.03837882 0.03684341 0.03619902 0.03733729 0.03630070 0.03410561
&  0.03438815 0.03237045 0.03261731 0.03416608 0.03121338 0.03201160
& 1  0
-0.10412377 0.03530452 0.04474088 0.04512217 0.03907324 0.03827783
&  0.03712461 0.03295426 0.04022175 0.04198539 0.04282273 0.04346007
&  0.04273191 0.04063471 0.04114667 0.04159701 0.04061975 0.03960966
&  0.03838484 0.03676783 0.03618738 0.03745855 0.03626942 0.03375115
&  0.03440080 0.03295662 0.03301495 0.03374968 0.03151317 0.03228719
& 1  0
-0.10238326 0.03561052 0.04382800 0.04512217 0.03885667 0.03808950
&  0.03706032 0.03154492 0.04067270 0.04226781 0.04287958 0.04363774
&  0.04266048 0.04086227 0.04122862 0.04159732 0.04095171 0.03949742
&  0.03846224 0.03720802 0.03634637 0.03713581 0.03636644 0.03465282
&  0.03455278 0.03187590 0.03242711 0.03448124 0.03125790 0.03120587
& 1  0
-0.10946020 0.03800291 0.04325401 0.04512217 0.04041234 0.03625963
&  0.03688524 0.03301763 0.03923185 0.04105163 0.04269354 0.04343358
&  0.04300986 0.04123552 0.04060520 0.04142916 0.04046444 0.04026002
&  0.03818432 0.03682291 0.03724765 0.03733290 0.03601997 0.03451568
&  0.03533661 0.03499694 0.03339726 0.03316743 0.03478948 0.03259249
& 1  0
-0.10096127 0.03773260 0.04376575 0.04512217 0.04026209 0.03566665
&  0.03687855 0.03194518 0.03832100 0.04069720 0.04248551 0.04321547
&  0.04264282 0.04096236 0.04040007 0.04120822 0.04033493 0.04010351
&  0.03792932 0.03681504 0.03730850 0.03770689 0.03614185 0.03445579
&  0.03460285 0.03467387 0.03311365 0.03210032 0.03420875 0.03163605
& 1  0
-0.09864677 0.03775632 0.04361893 0.04512217 0.03998230 0.03619930
&  0.03638698 0.03272732 0.03896292 0.04068225 0.04228354 0.04297513
&  0.04260976 0.04094456 0.04029595 0.04117116 0.04022218 0.03998305
&  0.03764655 0.03640016 0.03708801 0.03699700 0.03590523 0.03389567
&  0.03527287 0.03421544 0.03295929 0.03281600 0.03411650 0.03127680
& 1  0
```

TABLE 4.1A: NEURAL NETWORK TRAINING USING $10*\text{LOG}_{10}$ RULE

NEURAL NETWORK TRAINING FILE

(X1 pulses 7-8, X2 pulses 1-5)

```
-0.10379639 0.03806691 0.04415456 0.04512217 0.04000709 0.03647318
&  0.03696489 0.03343683 0.03845018 0.04084762 0.04240947 0.04324443
&  0.04274341 0.04100363 0.04063063 0.04133947 0.04052914 0.04021049
&  0.03822340 0.03681338 0.03679876 0.03699968 0.03580268 0.03461304
&  0.03450352 0.03442964 0.03333210 0.03269047 0.03410249 0.03131648
& 1  0
-0.09621912 0.03479013 0.04334402 0.04512217 0.03948681 0.03874464
&  0.03526762 0.03686932 0.03834803 0.04110362 0.04134914 0.04267878
&  0.04212548 0.03997838 0.04045517 0.04035744 0.03996985 0.03949738
&  0.03734601 0.03658093 0.03726464 0.03489060 0.03590131 0.03500549
&  0.03357158 0.03355487 0.03414739 0.03348790 0.03244893 0.03339234
& 1  0
-0.10472980 0.03569208 0.03490455 0.02944288 0.03592090 0.03765672
&  0.04257619 0.04398213 0.04063548 0.04188770 0.04291037 0.03833229
&  0.04468902 0.04583590 0.04631609 0.04619198 0.04408585 0.04086791
&  0.04135226 0.03925247 0.02610017 0.03534030 0.03471308 0.03318219
&  0.03507054 0.03352930 0.03359315 0.03591629 0.03298004 0.03177198
& 0  1
-0.11423284 0.03697584 0.03321140 0.02944288 0.03090771 0.03986090
&  0.04333232 0.04419208 0.04239541 0.04130497 0.04328245 0.04203018
&  0.04478324 0.04747504 0.04822262 0.04746034 0.04493113 0.04008196
&  0.04176842 0.04211552 0.03972958 0.03098337 0.03527343 0.03472697
&  0.02718298 0.03291009 0.03417993 0.03373240 0.03003015 0.03072631
& 0  1
-0.10647562 0.03411767 0.02921005 0.02944288 0.03827274 0.03834013
&  0.04237091 0.04465111 0.04149322 0.04188736 0.04339307 0.03932660
&  0.04518953 0.04597174 0.04656535 0.04666887 0.04453281 0.04099870
&  0.04136220 0.03829430 0.03629588 0.03677536 0.03079313 0.03439039
&  0.03334536 0.03229596 0.03388935 0.03299620 0.02678400 0.03044681
& 0  1
-0.10501991 0.03589363 0.03236945 0.02944288 0.03245705 0.03922029
&  0.04213724 0.04267701 0.04110792 0.04016834 0.04162046 0.04069242
&  0.04350005 0.04597550 0.04666584 0.04588801 0.04334099 0.03827481
&  0.04043083 0.04070264 0.03893869 0.03399264 0.03244048 0.03420498
&  0.03159380 0.03174811 0.03417691 0.03464519 0.03344666 0.03199372
& 0  1
-0.10180893 0.03706216 0.03608059 0.02944288 0.03372297 0.04123339
&  0.04368668 0.04421203 0.04319250 0.04105323 0.04108768 0.04191953
&  0.04331473 0.04468200 0.04499360 0.04404569 0.04142466 0.03772487
&  0.03886177 0.03855310 0.03577731 0.03014992 0.03337440 0.03435041
&  0.03432923 0.03367878 0.03062972 0.03107096 0.03250538 0.03222779
& 0  1
```

TABLE 4.1B: NEURAL NETWORK TRAINING USING 10*LOG$_{10}$ RULE

After training is completed, testing sets are produced with SNR levels from 0 dB to -20 dB using white Gaussian noise as outlined in Chapter 3. Appendix B presents the NN results for all **X*i*_*j*** tested against the trained network. There are 16 possible sets for **X1_*j*** and 5 sets for **X2_*j***. Each table in Appendix B presents separate NN outputs for each **X1$_t$** or **X2$_t$** with SNR values of 0 to -5 dB (in one dB steps), -10 dB, -15 dB and -20 dB. The notation **X*i*_*j*** is used to represent the template **X*i*** with -*j* dB SNR. For example X1_2dB means that each transient within the X1 template has an SNR of -2 dB. The X1 and X2 columns under the NN Ideal Output have either ones or zeros. A number one in the X1 column and a zero in the X2 column under the NN Ideal Output means that the set of values being tested are related to X1 and not X2. The X1 and X2 under the NN Tested Output represent the actual values obtained after testing. Numbers closest to 1 indicate strong correlation between the trained NN and the tested values, whereas numbers closest to zero indicate weak correlation between the trained NN and the tested values.

A summary of results for the noisy signals produced from a single run is presented in Table 4.2. Top and bottom sections represent testing using either **X1$_t$** or **X2$_t$** respectively. The first column in each section designates which test set is used. The second and fourth columns display average output values with a maximum possible value displayed on top of the column. For example, a value of 0.9887873 is obtained in the **X1** table under the "1" section in column two for X1_(-3 dB). This result demonstrates the NN has calculated a 0.9887873 out of a possible 1.0 value, indicating that the signal tested is more than likely a member of the **X1$_t$** class. The third and fifth columns display the standard deviation of numbers in obtaining each average value. These numbers are very useful in determining how close each average value is in relation to the overall average. Standard deviation values are a big determination factor in choosing to use each coefficient expressed in dB during the training process. Recall that the sigmoid transfer function is used in the NN configuration. By definition, the results should then be in the interval [0,1] however, the *NeuralWorks Professional II/Plus* software internally rescales the output values when using the MiniMax table option. For this reason some output values may be slightly larger than 1 or slightly less than zero.

X1

	1		0	
Tested Sets	**Avg.**	**Std. Dev.**	**Avg.**	**Std. Dev.**
X1_(0dB)	1.0047611	0.0230146	-0.0017851	0.0223812
X1_(-1dB)	1.0030456	0.0254554	-0.0000231	0.0232048
X1_(-2dB)	0.9923662	0.0238875	0.0109496	0.0240785
X1_(-3db)	0.9887873	0.0375286	0.0141106	0.0385814
X1_(-4dB)	0.9837167	0.0585917	0.0194570	0.0573308
X1_(-5dB)	0.9757227	0.0320483	0.0281420	0.0317096
X1_(-10dB)	0.9600819	0.0503517	0.0452866	0.0487668
X1_(-15dB)	0.7987847	0.1200738	0.2080957	0.1236332
X1_(-20dB)	0.8242014	0.1231212	0.1804250	0.1222050

X2

	0		1	
Tested Sets	**Avg.**	**Std. Dev.**	**Avg.**	**Std. Dev.**
X2_(0dB)	0.2303900	0.1245466	0.7722318	0.1229367
X2_(-1dB)	0.2271358	0.0718728	0.7719382	0.0672361
X2_(-2dB)	0.3569218	0.1859071	0.6483716	0.1830204
X2_(-3db)	0.2896448	0.1613317	0.7211064	0.1562198
X2_(-4db)	0.3098798	0.1032637	0.6901816	0.1072366
X2_(-5dB)	0.3407796	0.2344198	0.6639332	0.2357792
X2_(-10dB)	0.6156368	0.1799645	0.3885900	0.1677991
X2_(-15dB)	0.6786284	0.1034576	0.3235496	0.0986601
X2_(-20dB)	0.7075180	0.1573530	0.3078052	0.1540503

TABLE 4.2: AVERAGE VALUES FROM SINGLE NN TESTING RUN

Results for the single run are very good. The X1 portion of Table 4.2 demonstrates correlation values of 0.97 or greater down to -5 dB SNR and a strong performance of 0.82 down to -20 dB with a very low standard deviation. The X2 portion of Figure 4.2 displays results that are not quite as good. A 0.72 value was obtained at -3 dB. This may be due to the increased diversity amongst transients in the X2 set.

2. Multiple Runs

Once results are obtained for a single run case, multiple runs are completed. The intent of multiple runs is to test the integrity of the trained NN by obtaining many sets of outputs and to have statistically more reliable results. The MATLAB® software package is used once again to generate all SNR data files.

Table 4.3 represents the average values calculated for 100 runs. The results are organized in the same fashion as in Figure 4.2.

X1

	1		0	
Tested Sets	**Avg.**	**Std. Dev.**	**Avg.**	**Std. Dev.**
X1_(0dB)	0.9888676	0.0426854	0.0134650	0.0347443
X1_(-1dB)	0.9901707	0.0348740	0.0128657	0.0347769
X1_(-2dB)	0.9855116	0.0410708	0.0179122	0.0409456
X1_(-3db)	0.9847976	0.0419061	0.0187440	0.0417753
X1_(-4dB)	0.9800479	0.0453721	0.0238660	0.0455406
X1_(-5dB)	0.9759416	0.0494418	0.0280618	0.0494439
X1_(-10dB)	0.9443232	0.0718501	0.0612471	0.0725386
X1_(-15dB)	0.8786200	0.1017792	0.1285275	0.1026012
X1_(-20dB)	0.8139043	0.1259401	0.1949490	0.1267664

X2

	0		1	
Tested Sets	**Avg.**	**Std. Dev.**	**Avg.**	**Std. Dev.**
X2_(0dB)	0.2251065	0.1118897	0.7790914	0.1108506
X2_(-1dB)	0.2339048	0.1115452	0.7700723	0.1116177
X2_(-2dB)	0.2802127	0.1233627	0.7236593	0.1229222
X2_(-3db)	0.2960157	0.1390701	0.7087087	0.1386071
X2_(-4db)	0.3342565	0.1446113	0.6701185	0.1442858
X2_(-5dB)	0.3669860	0.1446713	0.6374069	0.1450339
X2_(-10dB)	0.5174339	0.1599166	0.4871487	0.1612239
X2_(-15dB)	0.6482288	0.1568769	0.3581118	0.1574939
X2_(-20dB)	0.7228622	0.1361579	0.2866304	0.1371880

TABLE 4.3: AVERAGE VALUES FROM 100 CONSECUTIVE NN TESTING RUNS

Results show that the average correlation values under the "1" section of the **X1** portion of Tables 4.2 and 4.3 are greater than the values in the "1" section of the **X2** portion. These values are a function of the types of transients within the data sets used. High correlation values in the **X1** part of Tables 4.2 and 4.3 are due to the small amount of diversity among transients composing the X1 set. Whereas the lower correlation results in the **X2** portion of Tables 4.2 and 4.3 are attributed to the increased diversity between transients composing **X2**.

V. SOFTWARE IMPLEMENTATION

Two types of software packages are used in this study, MATLAB® [4] from Math Works Inc., and NeuralWorks Professional II/Plus™ from NeuralWare, Inc.

A. NEURALWORKS PROFESSIONAL II/PLUS™

All neural network (NN) programming presented in Chapter IV is performed using NeuralWorks Professional II/Plus™ from NeuralWare, Inc.. System training and testing is done as stated in Chapter 4. "Snapshots" of the network architecture in untrained, training and trained mode are presented in Figures B.39, B.40, B.41 and B.42 in Appendix B.

B. MATLAB® SOFTWARE IMPLEMENTATION

All spectral measure studies presented in Chapters II and III are completed by using the MATLAB® program package to take advantage of the software graphing capabilities [4]. The training and test files for the NN are also produced using MATLAB®.

Several programs and functions are written to handle single and multiple run spectral measures. There are two main program sets. Each program set operates on sponsor supplied raw data sets the same way. The major difference between the single and multiple run algorithms are the linking of all programs in order to repeat runs. All programs in the multiple run case are linked sequentially, producing output values which are stored in matrix form for averaging and plotting.

Both sets of programs utilize two basic functions, LOFAR.M and DIST.M. Either program may be used with or without degraded signal sets. A larger program entitled "BIG.M" was written to incorporate LOFAR.M, DIST.M, noise generation functions, summing functions, averaging functions and plotting routines into one large recursive algorithm in order to accomplish multiple runs. All MATLAB® code used in this work is presented in Appendix C.

1. Program DIST.M [1]

The program DIST.M was modified from [1]. This program computes all frequency-based measures presented in Chapters II and III. In addition, this program also computes the various frequency-based measure averages and standard deviations for each of the files tested. The reference file is designed to be the reference template (i.e. $\mathbf{X1}_t$, $\mathbf{X2}_t$ or $\mathbf{X3}_t$), whereas the test files could be any set of spectral coefficients of interest.

2. Program LOFAR.M [1]

The program LOFAR.M was modified from [1]. This program computes the spectral coefficients used for the frequency-based measure and NN studies.

Several options exist in this program for raw data calculation. The first option is to truncate decimate/interpolate the data to compensate for the change in sampling rates. The remaining options are: mean removal, power spectrum and rectangular or Hamming window. The number of non-zero data points in the transform is 1020, the step size in data points is 1020 and the transform length is 1024.

VI. SUMMARY AND RECOMMENDATIONS

A. SUMMARY

This study examines spectral-based techniques to efficiently and reliably detect and categorize certain classes of transient signals embedded in additive white Gaussian noise. Two approaches have been considered.

The first approach considers the use of four spectral measures to distinguish between classes of transients. These measures include the normalized cross-correlation coefficient, the modified normalized cross-correlation coefficient, the Bhattacharyya distance and the divergence (related to the Kullback-Liebler number). Results show that these measures adequately classify the transients into three different classes. They also indicate that the modified normalized cross-correlation coefficient tends to have more robust classification features. Next, the robustness of the spectral-based measure to classify the transients is investigated by degrading the test data with additive white Gaussian noise. The SNRs used are between 0 dB and -20 dB. Results show that the modified normalized spectral cross-correlation coefficient (i.e. mean removed) exhibits the best performance.

The second approach uses a back-propagation NN approach. The NN was initially trained to distinguish between two different classes; $\mathbf{X1_t}$ and $\mathbf{X2_t}$. The third class $\mathbf{X3_t}$ is considered a compilation of different transient types, and is not used for training. Experimental results show that the NN can be successfully trained to distinguish between two classes under high and low SNR conditions.

The two problems with the NN implementation are a lengthy training time and classification of transients not belonging to the sets being trained on. Training is time consuming and is difficult to do accurately for small class sizes.

The spectral-based measure is based on inner products and is computationally faster. In addition, the NN classifies $X3_t$ as belonging to $X1_t$ or $X2_t$. Reducing the number of training iterations does not substantially improve the NN performance for this condition. The spectral measure techniques tend to classify unknown transients more reliably.

There are two drawbacks to both the spectral-based measure implementation and NN implementation. The first is the manual selection needed to define each reference template. This process is tedious and time consuming. However, once completed, the classification is fast for both the spectral measure techniques and the NN approach. The second drawback is the definition of the threshold used to decide whether or not a test transient belongs or does not belong to a specific class. Results show that the modified cross-correlation coefficient seems to be approximately 0.95 with no added noise and 0.4 at -10 dB SNR. Additional experiments with a larger number of transients per class are needed to accurately determine an appropriate threshold for detection.

B. RECOMMENDATIONS

There are three recommendations:

1.) The vector size is fixed at taking the first 64 or 30 spectral coefficients for the correlation-based and NN-based implementation, respectively in experiments. It is noted that a smaller number of spectral coefficients could be utilized for some of the transients studied. Reducing the number of power coefficients in the classification scheme will reduce the computational load. Thus one should examine the issues of potential speedup versus classification.

2.) Empirically the threshold used to decide whether or not a particular signal belongs to a given class seems to be a number around 0.95 for high SNR (essentially no added noise) for individual frequency-based measures. In order to obtain more accurate results at low SNRs, additional experiments with a larger numbers of transients per class are needed to accurately determine an appropriate threshold for classification in order to simulate real world results.

3.) The NN was trained using $\mathbf{X1_t}$ and $\mathbf{X2_t}$. The set $\mathbf{X3_t}$ was not used for training because of its odd transient composition. When $\mathbf{X3_t}$ was tested as an unknown transient group against the trained NN, results indicate that the NN is unable to determine that $\mathbf{X3_t}$ doesn't belong either to $\mathbf{X1_t}$ or $\mathbf{X2_t}$. Thus issues dealing with the relationships between small training set sizes, and unknown transients need to be investigated to help solve this problem.

REFERENCES

[1] Fargues, Monique P. and Hippenstiel, Ralph, *Investigation of Spectral-Based Techniques for Classification of Wideband Transient Signals*, Technical Report No. NPSEC-93-008, March 30, 1993, Naval Postgraduate School, Monterey, California.

[2] Strum, Robert D. and Kirk, Donald E., *Discrete Systems and Digital Signal Processing,*. Addison-Wesley Publishing Company, 1989.

[3] Therrien, C.W., *Decision Estimation and Classification. An Introduction to Pattern Recognition and Related Topics,* John Wiley & Sons, Inc. 1989 .

[4] *MATLAB® for the Macintosh, version 1.2c,* The MathWorks, Inc., May 6, 1991.

[5] Couch II, Leon W., *Digital and Analog Communication Systems, Fourth Edition,* Macmillan Publishing Company, 1993.

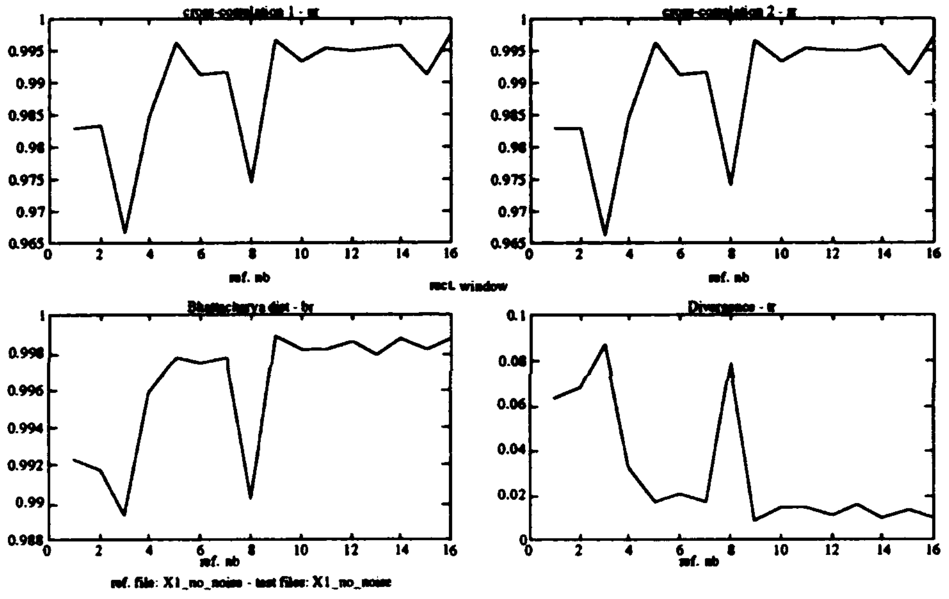
[6] Peebles, Peyton Z., *Probability, Random Variables, and Random Signal Principles,* McGraw-Hill, Inc., 1987.

[7] Technical Publishing Group, *Using NeuralWorks,* NeuralWare, Inc., 1993

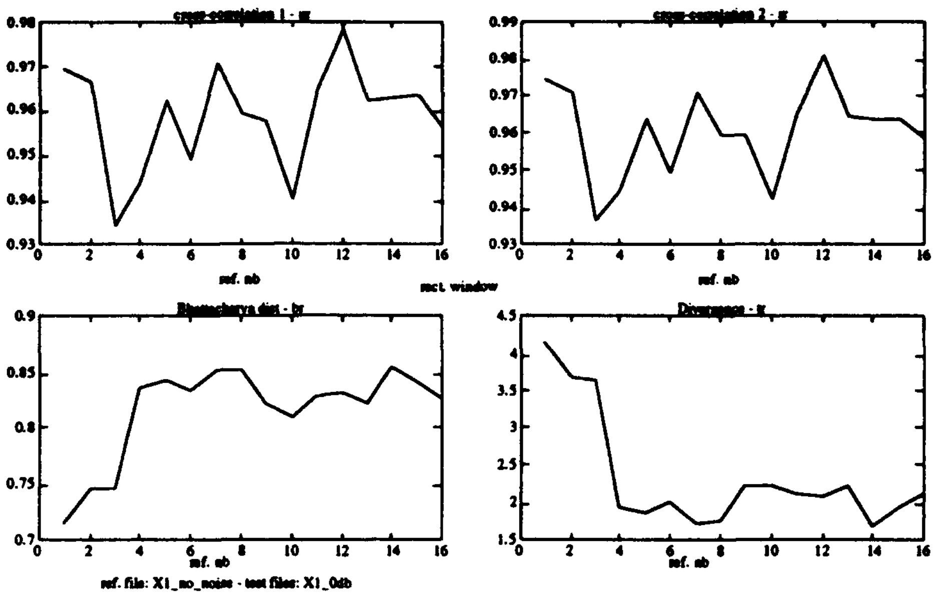
[8] Fargues, Monique P., *Unpublished Report,* February 1993, Naval Postgraduate School, Monterey, California.

APPENDIX A. INDIVIDUAL SPECTRAL MEASURES

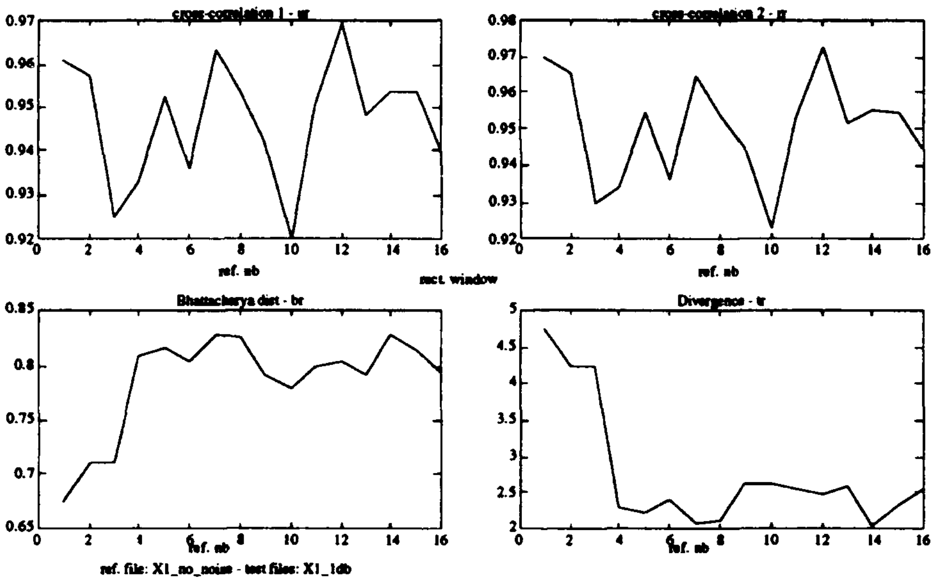
The following figures represent 60 individual spectral measure plots for each template as compared to other templates with varying SNRs. The lower left-hand portion of the graph displays the names of the two files tested. The first file is the reference template and the second file is the reference template with added noise.



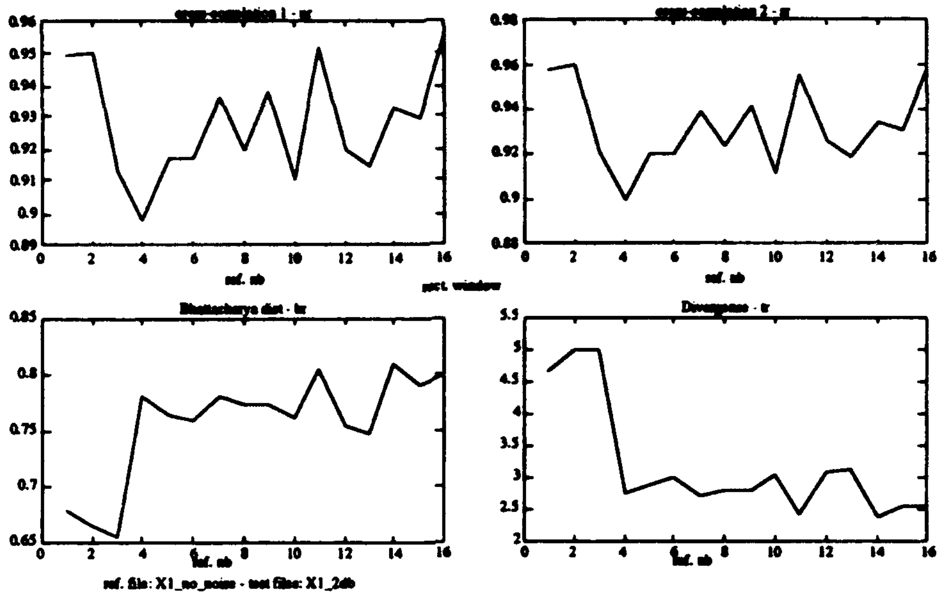
A.1: $X1_t$ correlated against X1



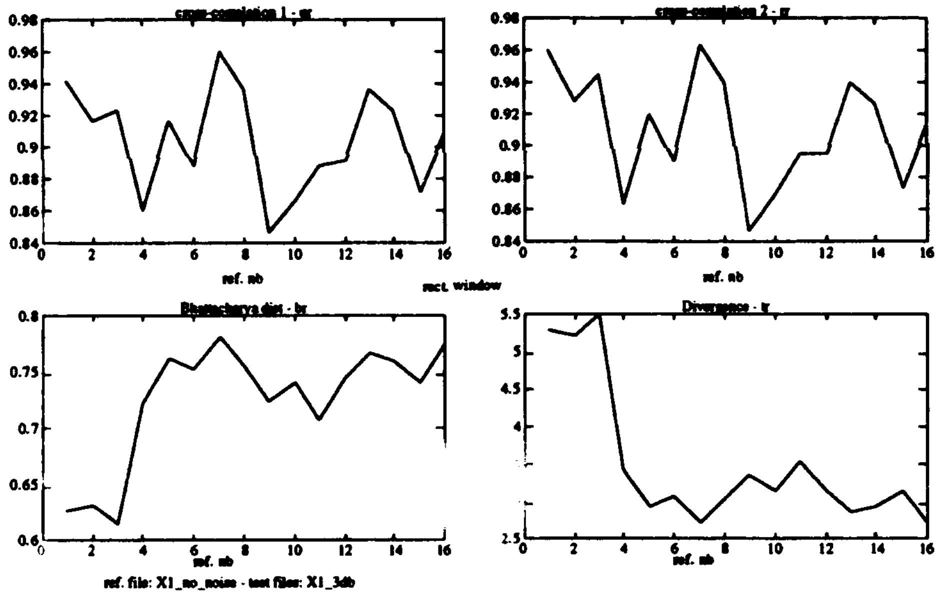
A.2: X1$_t$ correlated against X1_0 dB



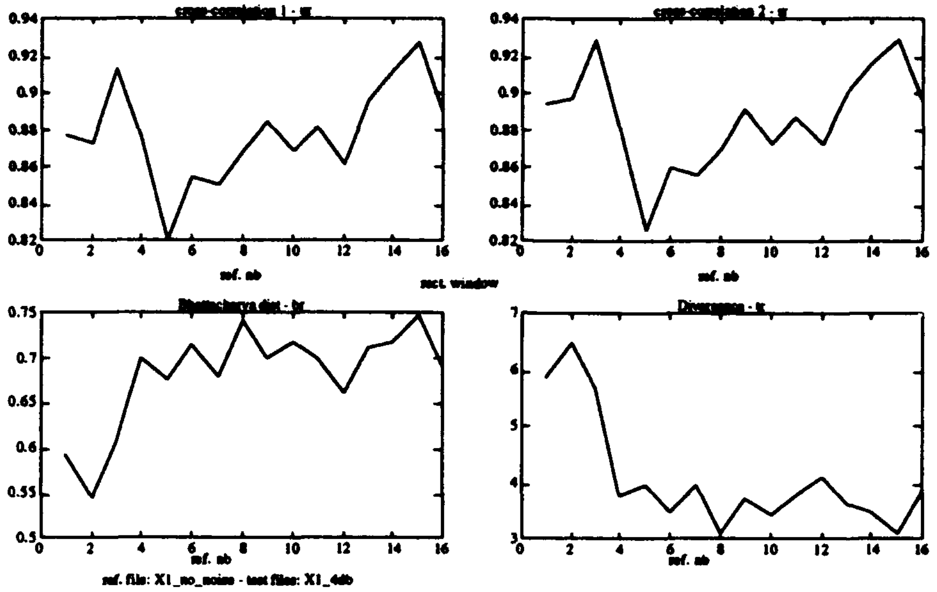
A.3: X1$_t$ correlated against X1_(-1dB)



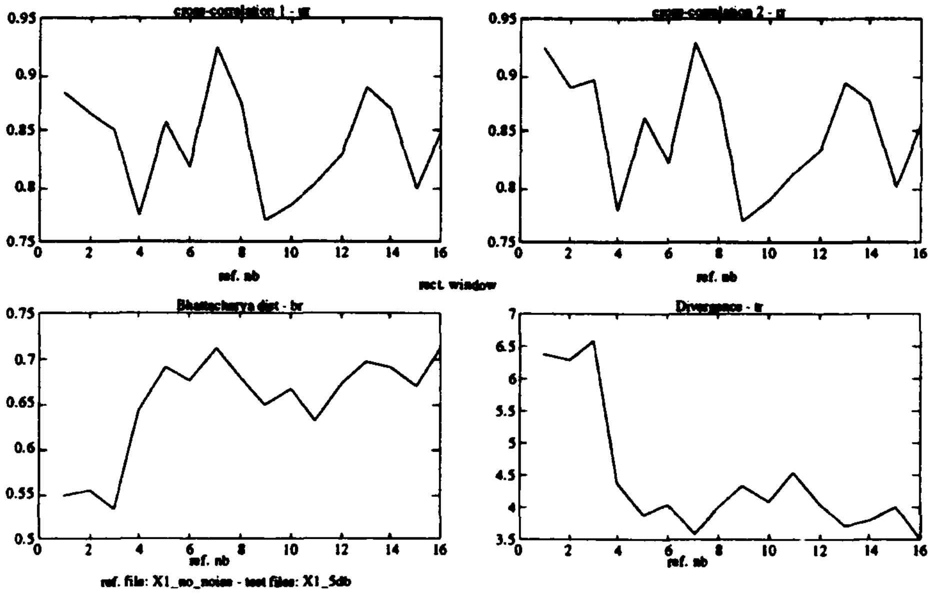
A.4: X1$_t$ correlated against X1_(-2dB)



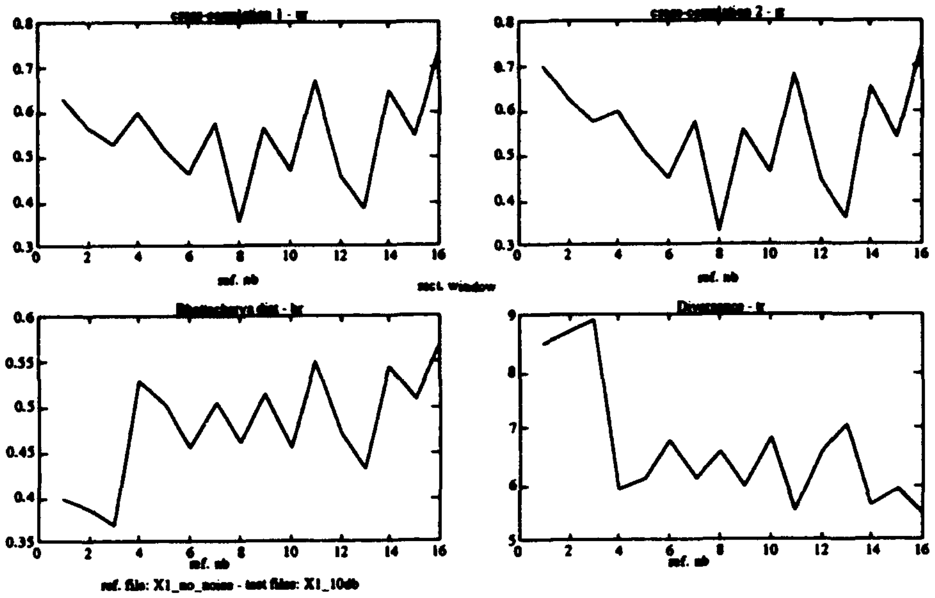
A.5: X1$_t$ correlated against X1_(-3dB)



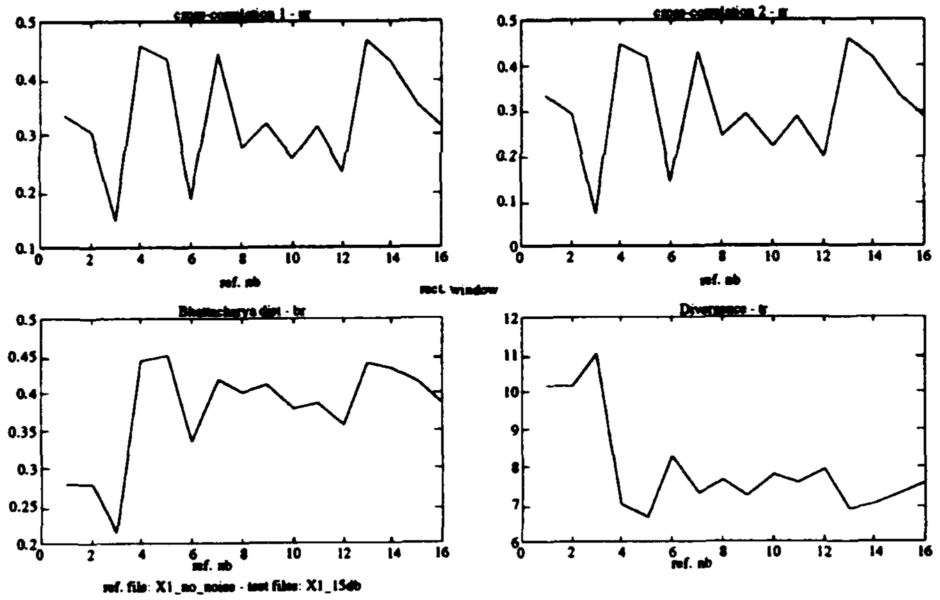
A.6: X1$_t$ correlated against X1_(-4dB)



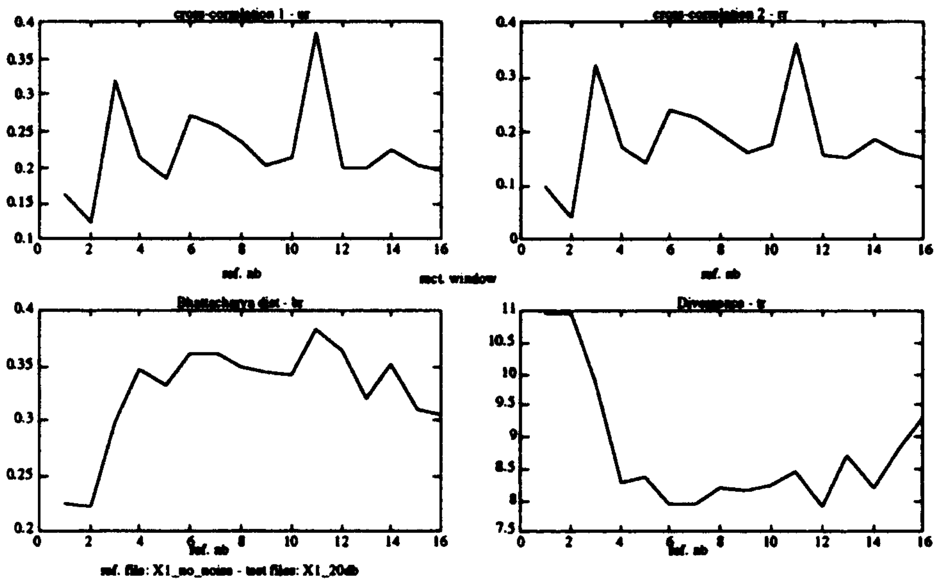
A.7: X1$_t$ correlated against X1_(-5dB)



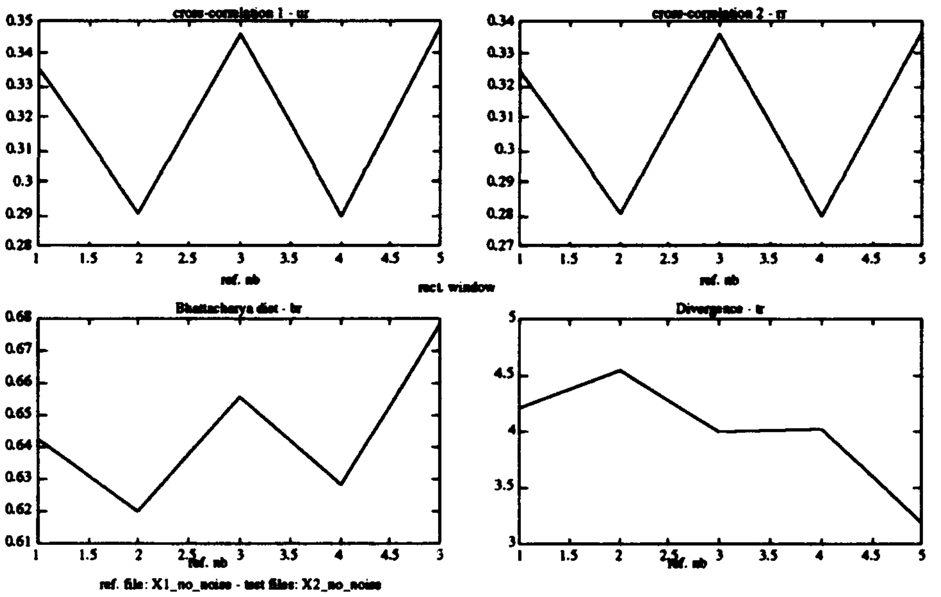
A.8: X1$_t$ correlated against X1_(-10dB)



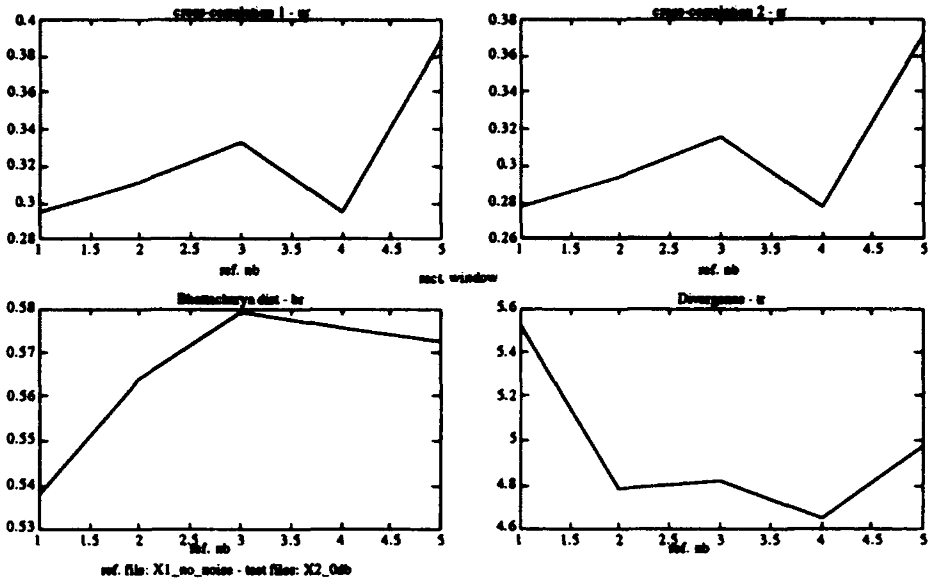
A.9: X1$_t$ correlated against X1_(-15dB)



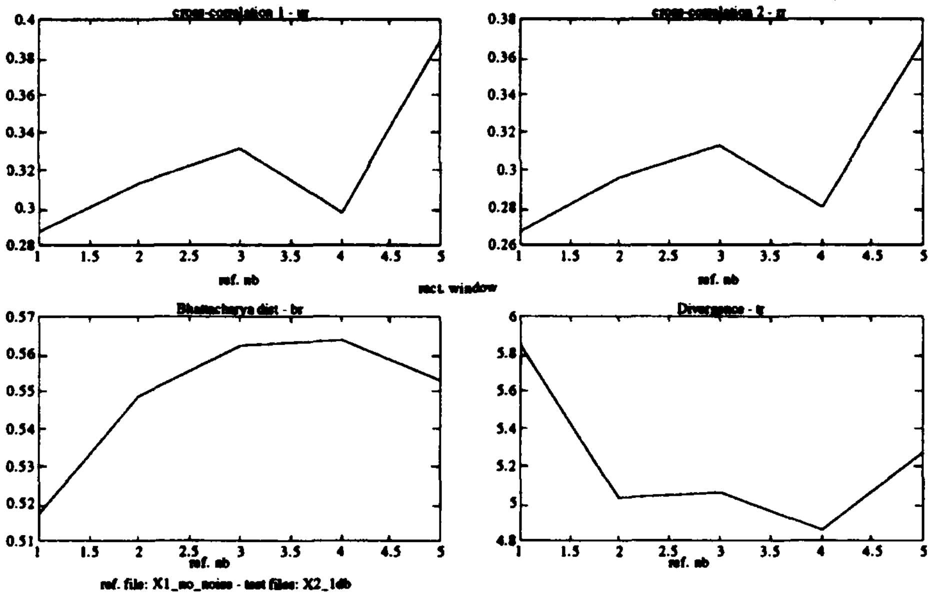
A.10: X1$_t$ correlated against X1_(-20dB)



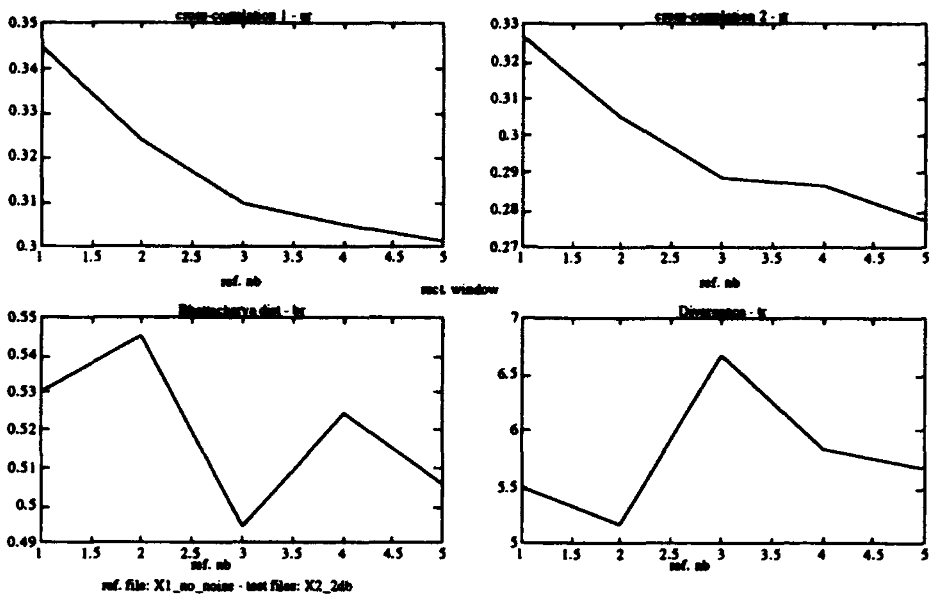
A.11: X1$_t$ correlated against X2



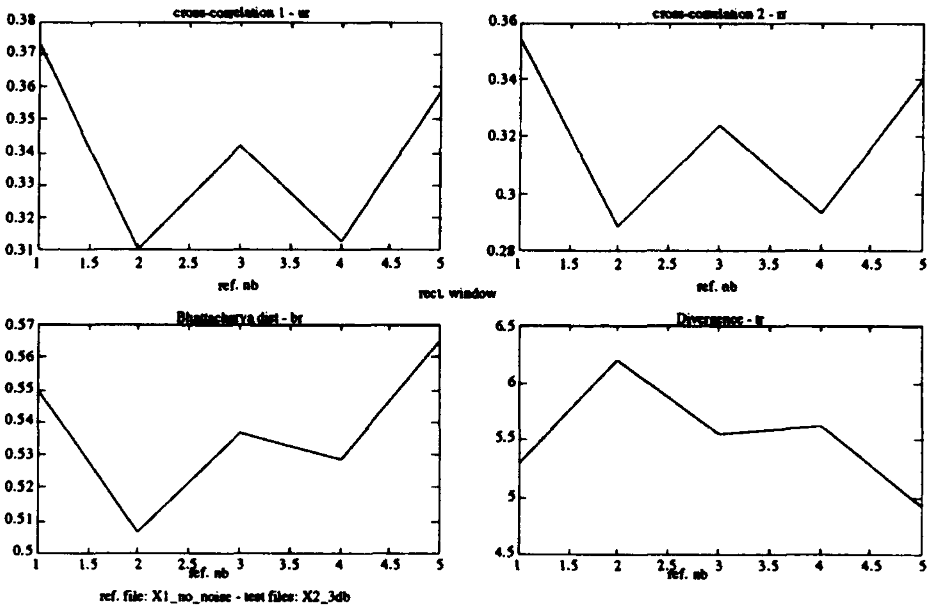
A.12: X1t correlated against X2_(0dB)



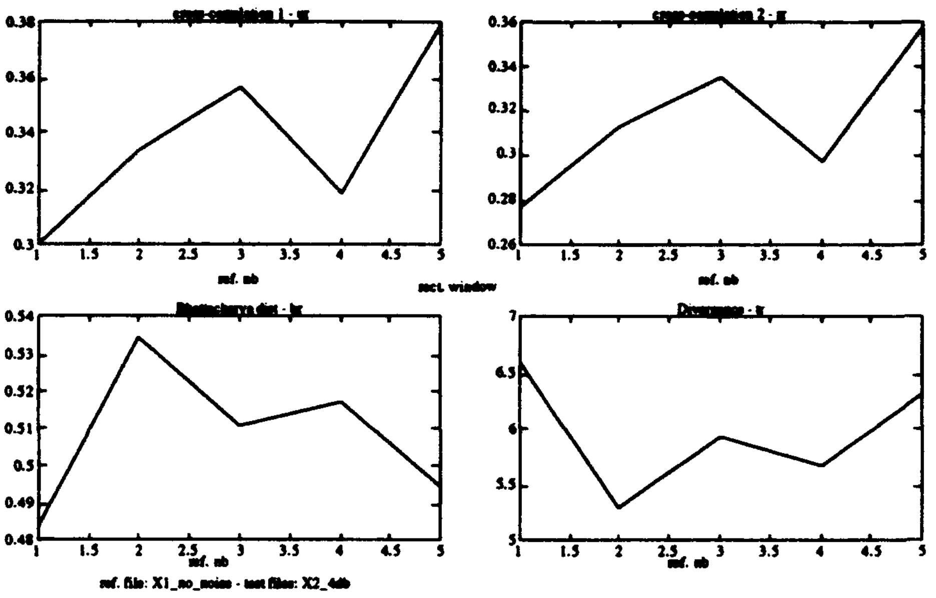
A.13: X1t correlated against X2_(-1dB)



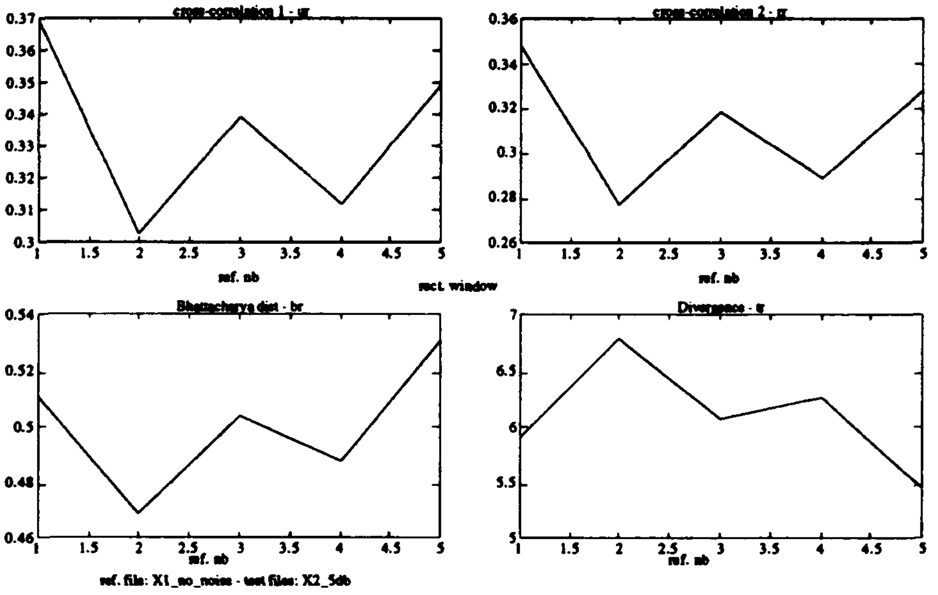
A.14: X1$_t$ correlated against X2_(-2dB)



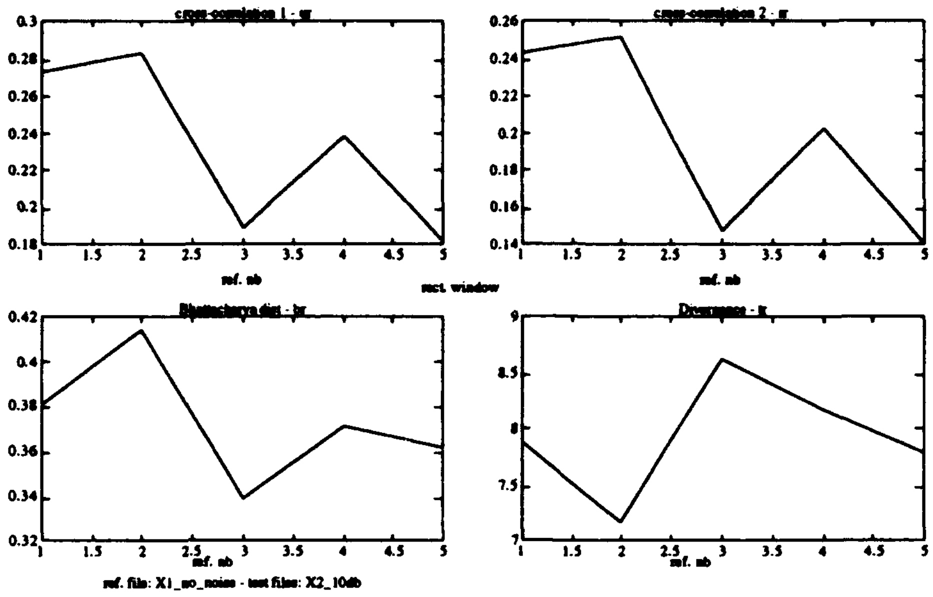
A.15: X1$_t$ correlated against X2_(-3dB)



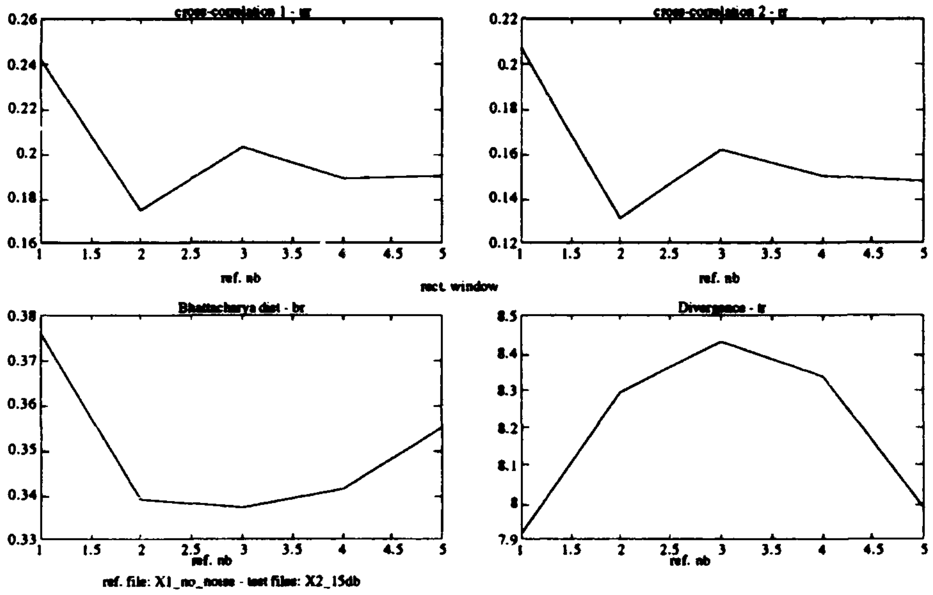
A.16: X1$_t$ correlated against X2_(-4dB)



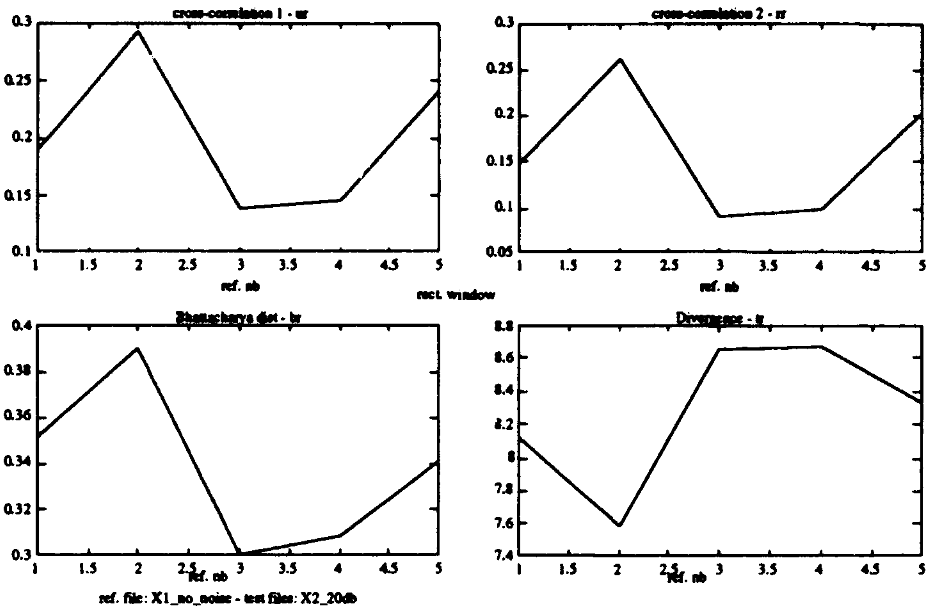
A.17: X1$_t$ correlated against X2_(-5dB)



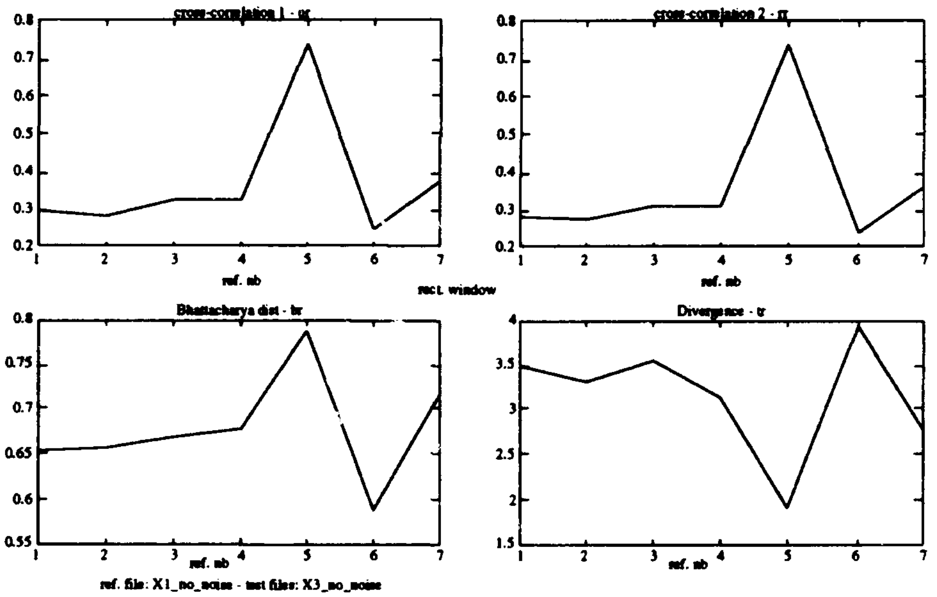
A.18: X1$_t$ correlated against X2_(-10dB)



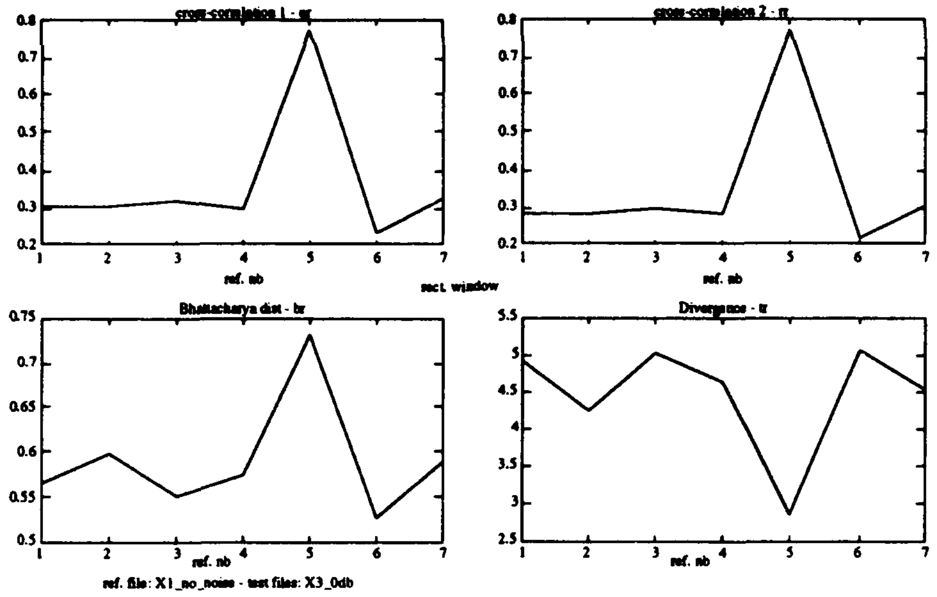
A.19: X1$_t$ correlated against X2_(-15dB)



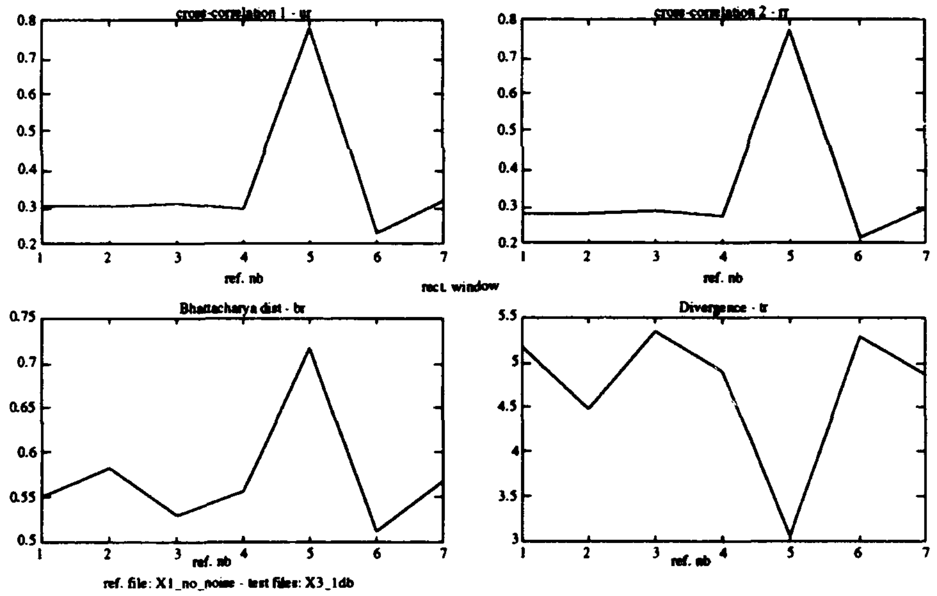
A.20: X1$_t$ correlated against X2_(-20dB)



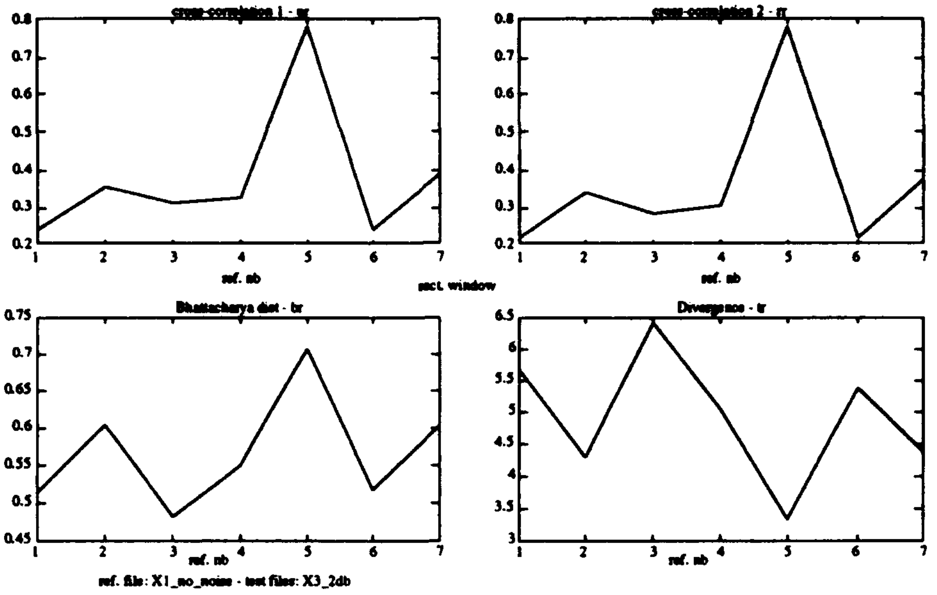
A.21: X1$_t$ correlated against X3



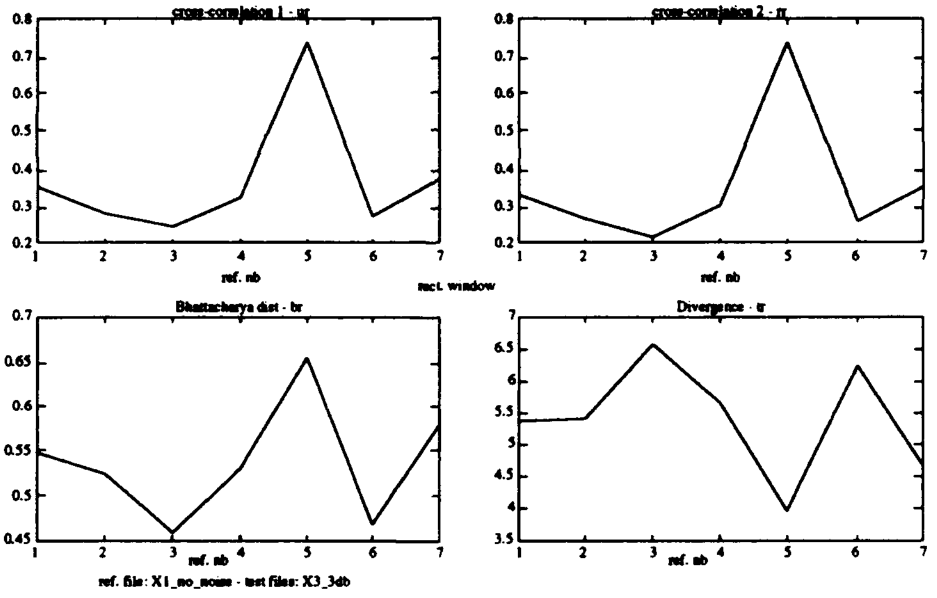
A.22: X1$_t$ correlated against X3_0dB



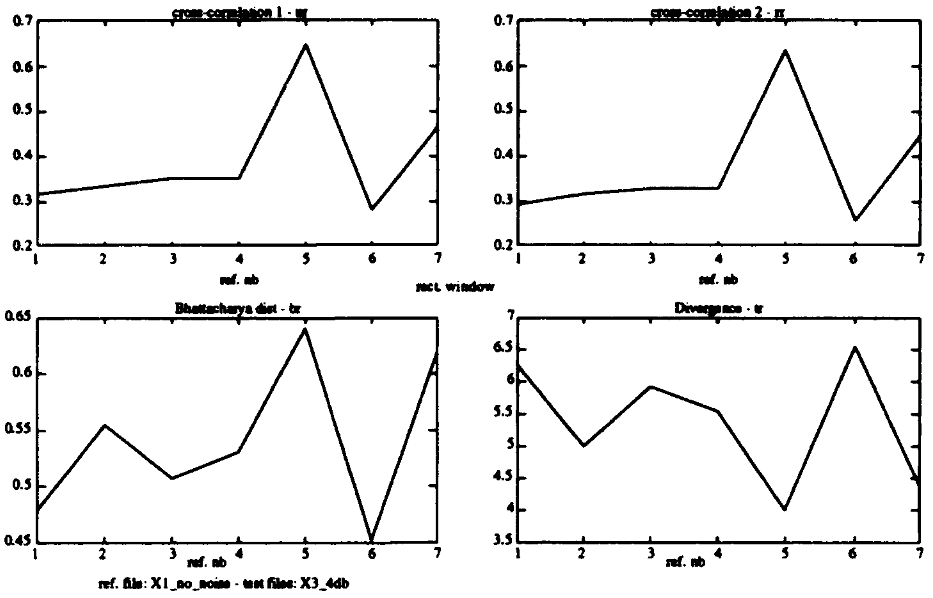
A.23: X1$_t$ correlated against X3_(-1dB)



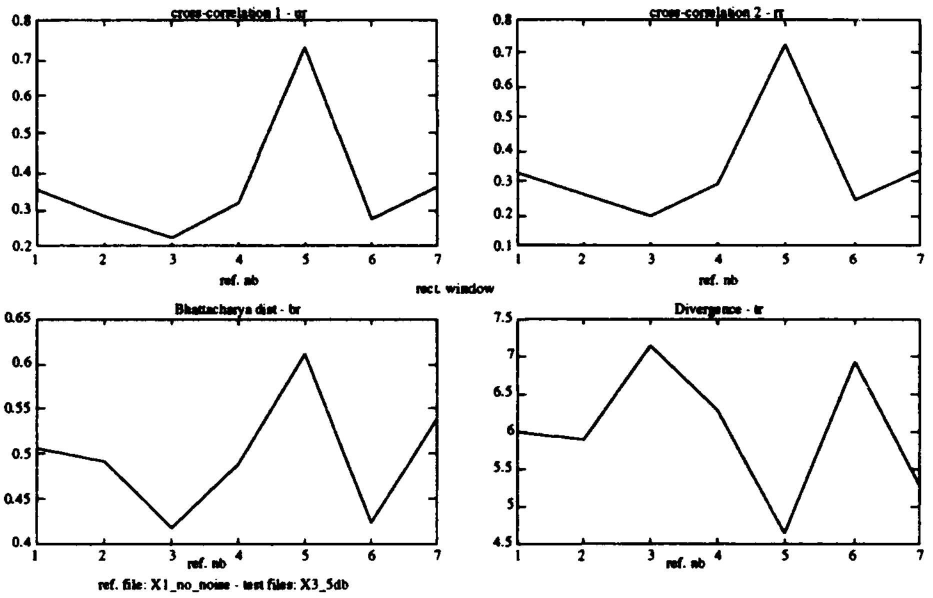
A.24: X1$_t$ correlated against X3_(-2dB)



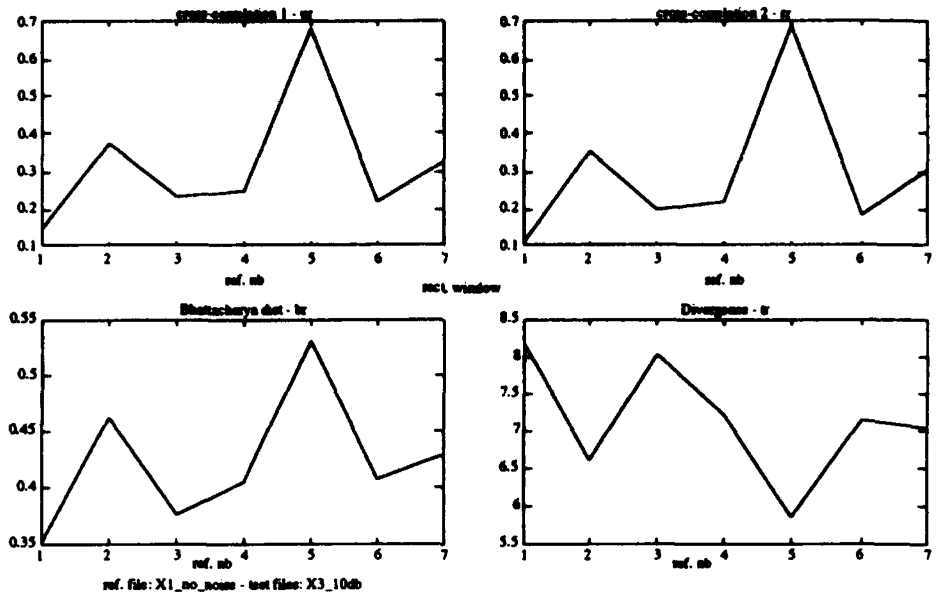
A.25: X1$_t$ correlated against X3_(-3dB)



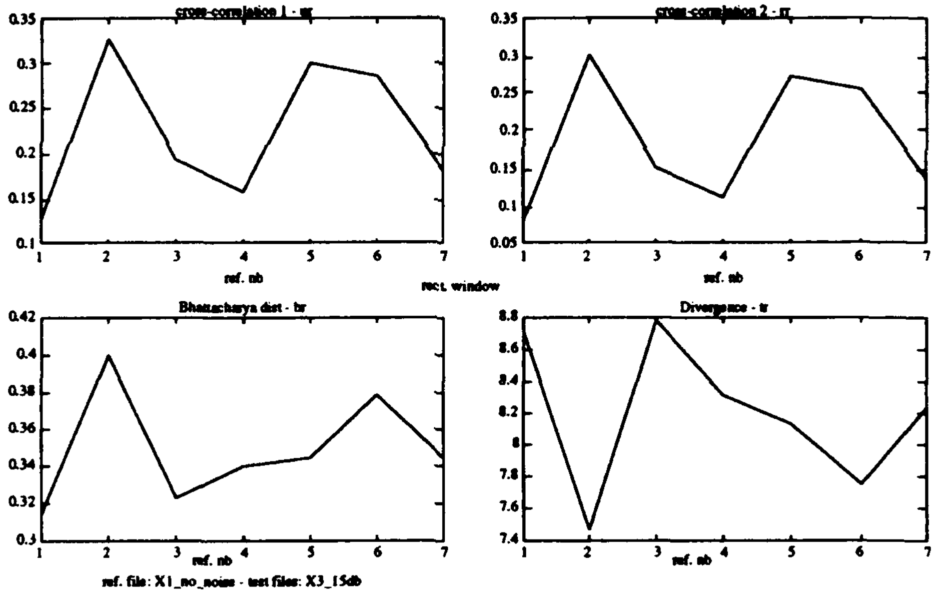
A.26: X1t correlated against X3_(-4dB)



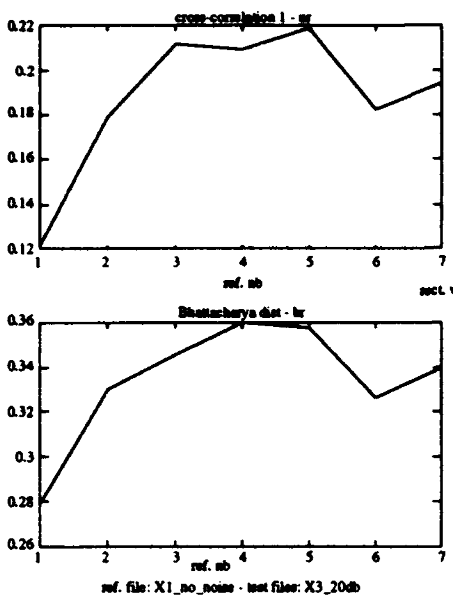
A.27: X1t correlated against X3_(-5dB)



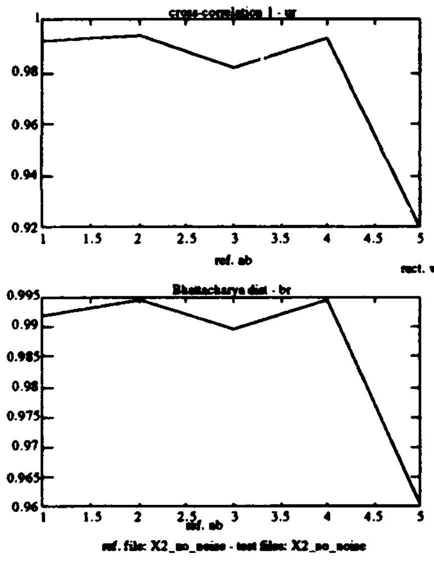
A.28: X1$_t$ correlated against X3_(-10dB)



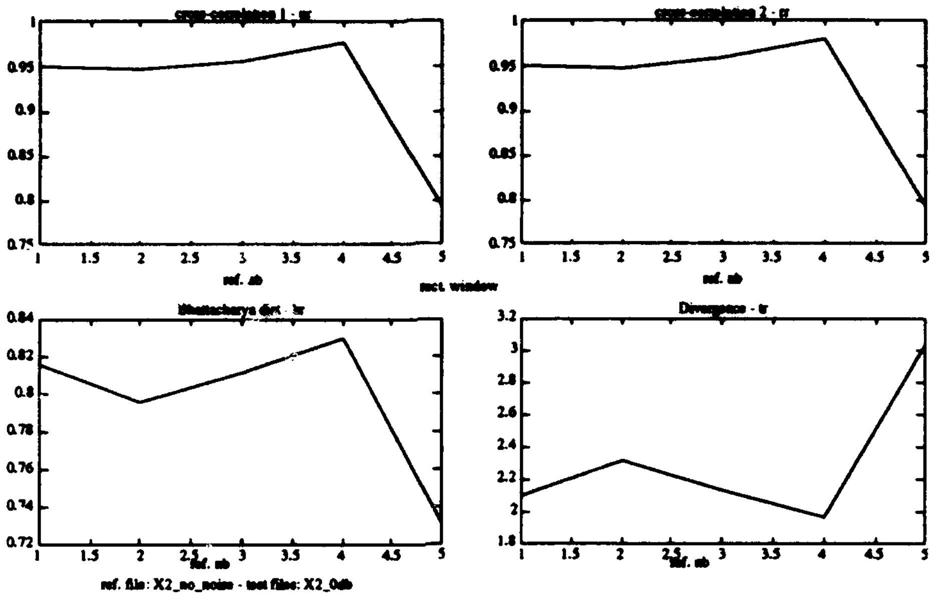
A.29: X1$_t$ correlated against X3_(-15dB)



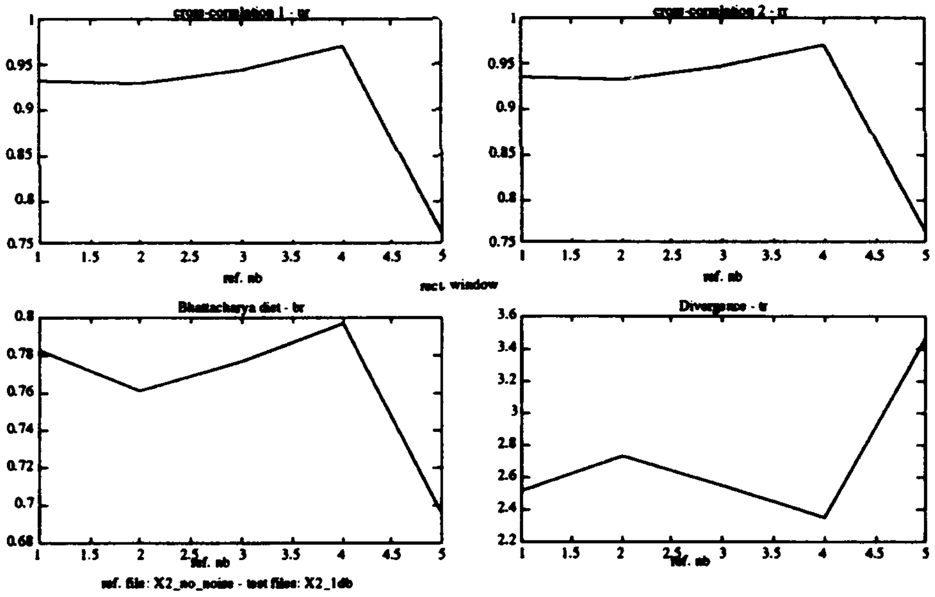
A.30: X1$_t$ correlated against X3_(-20dB)



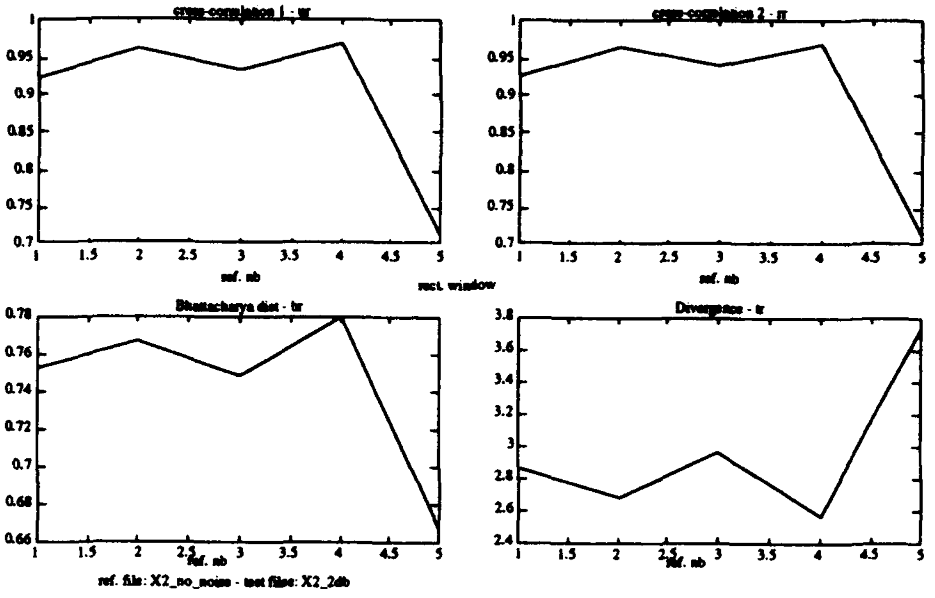
A.31: X2$_t$ correlated against X2



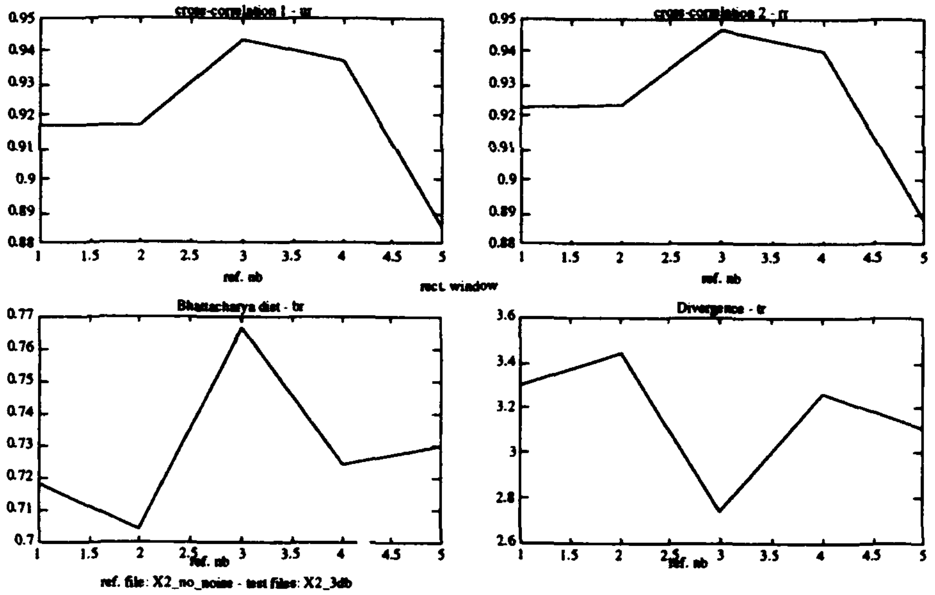
A.32: X2$_t$ correlated against X2_(0dB)



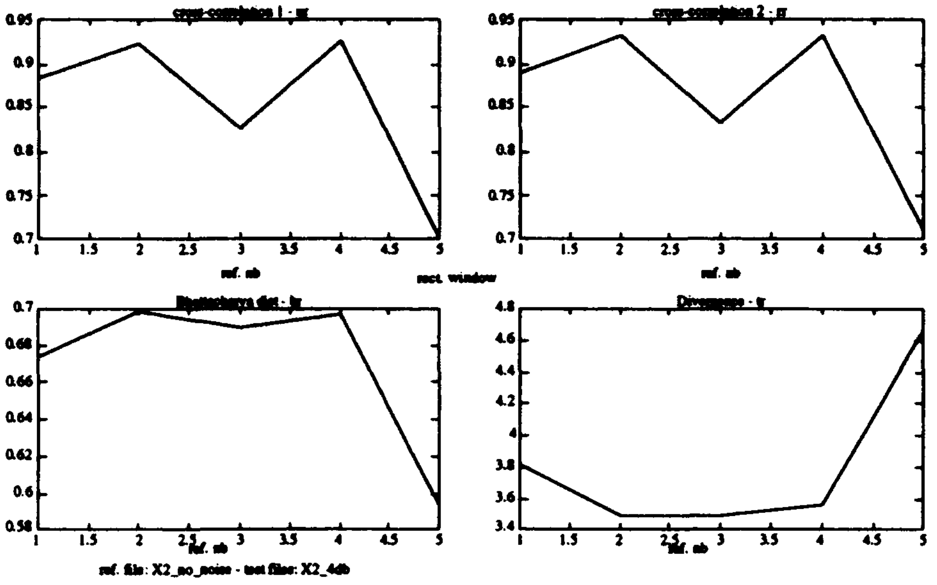
A.33: X2$_t$ correlated against X2_(-1dB)



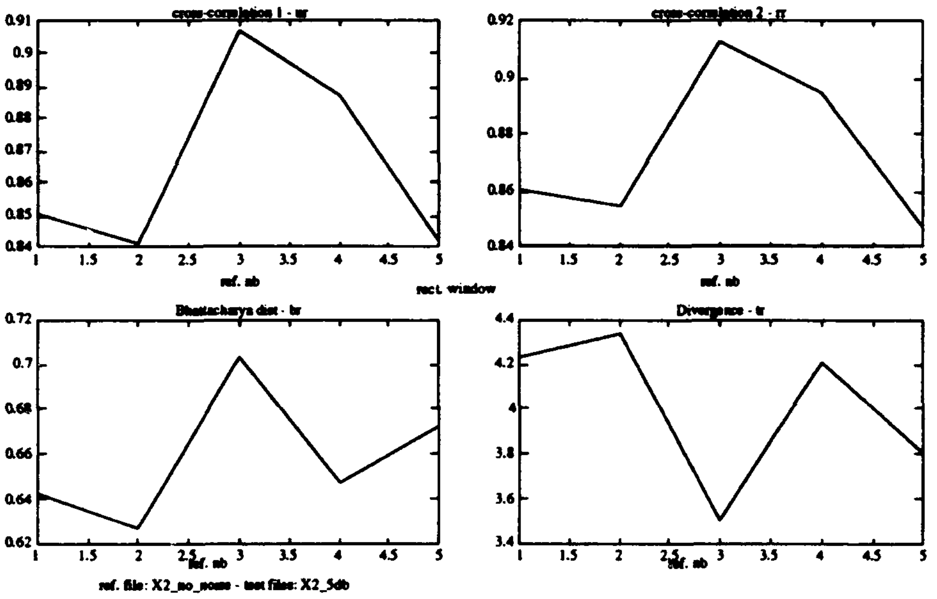
A.34: $X2_t$ correlated against X2_(-2dB)



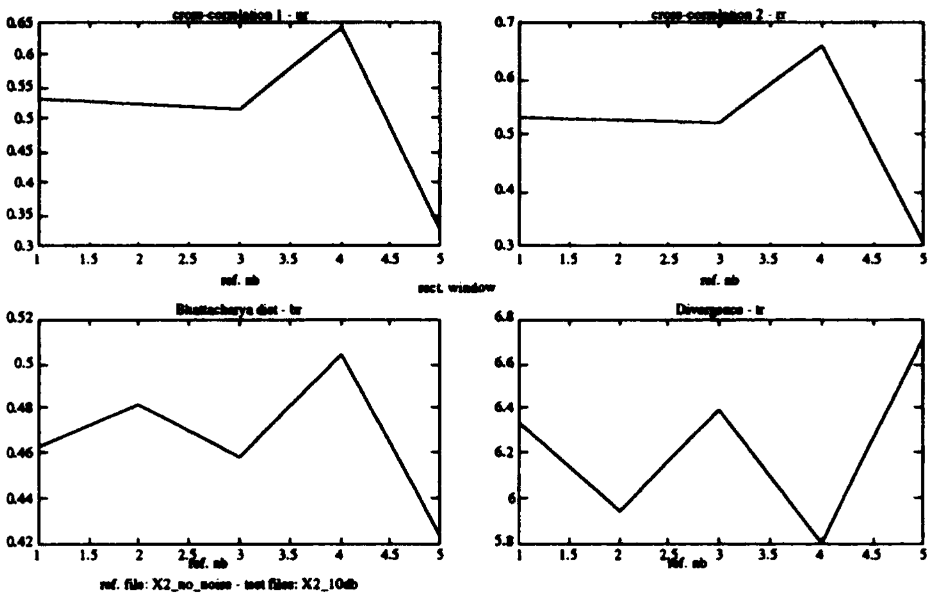
A.35: $X2_t$ correlated against X2_(-3dB)



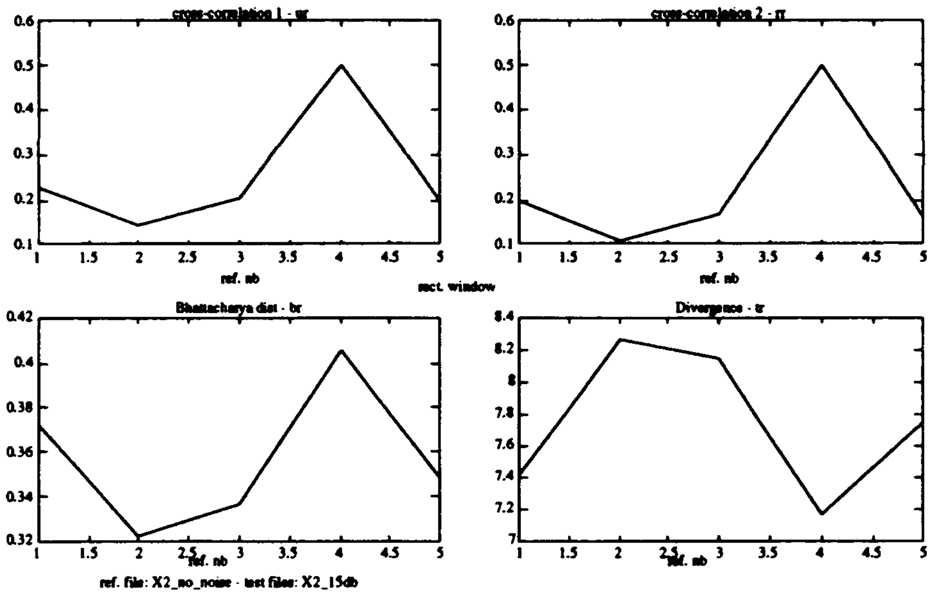
A.36: X2$_t$ correlated against X2_(-4dB)



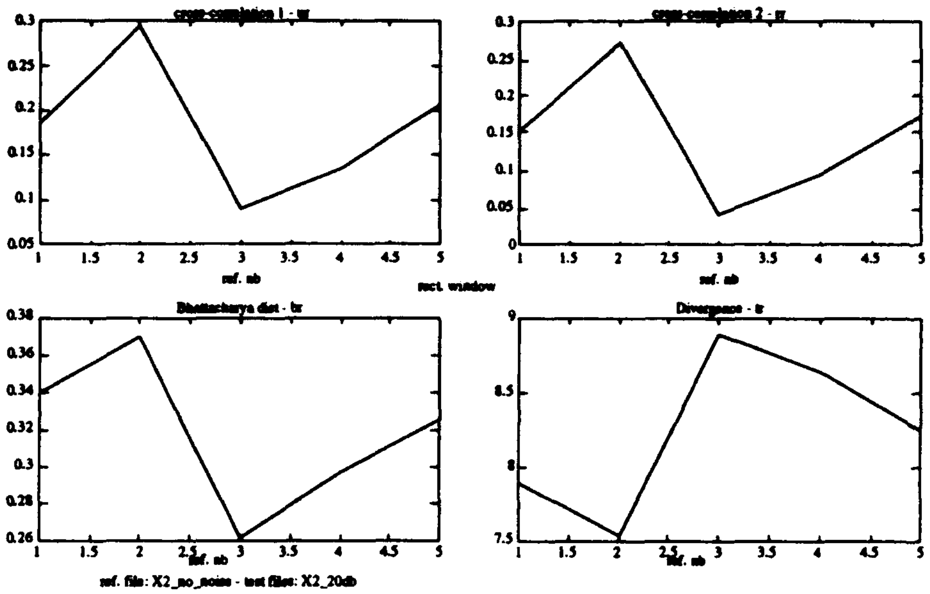
A.37: X2$_t$ correlated against X2_(-5dB)



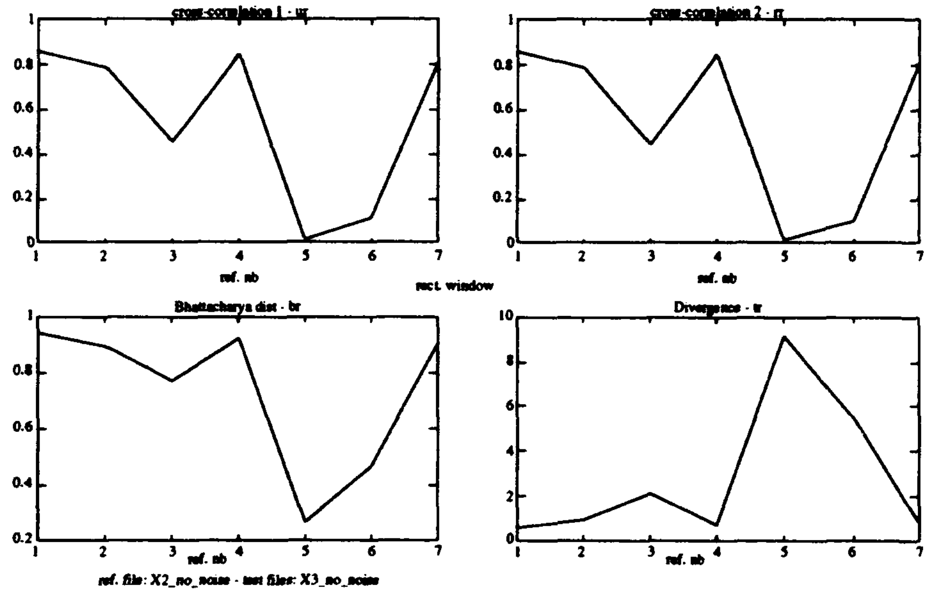
A.38: X2$_t$ correlated against X2_(-10dB)



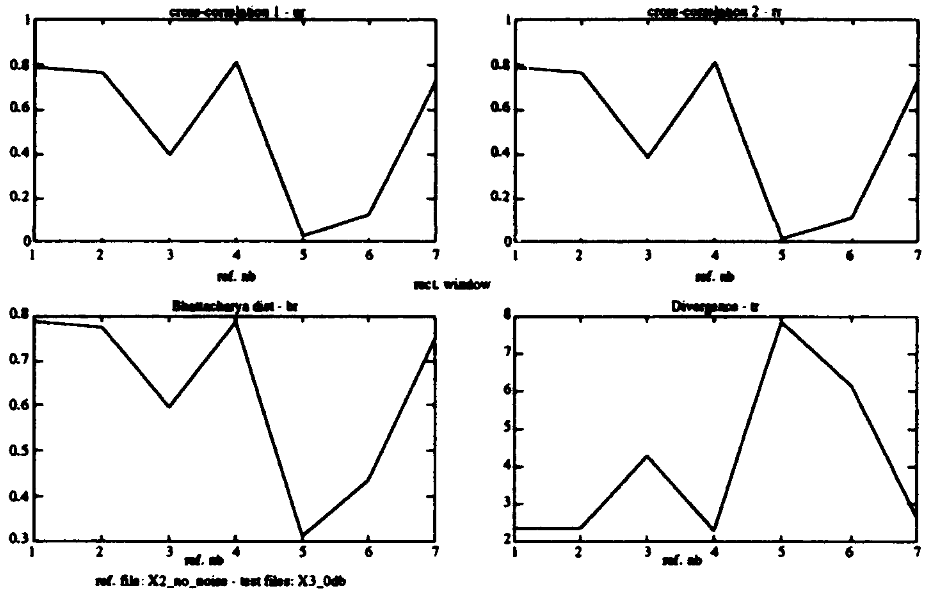
A.39: X2$_t$ correlated against X2_(-15dB)



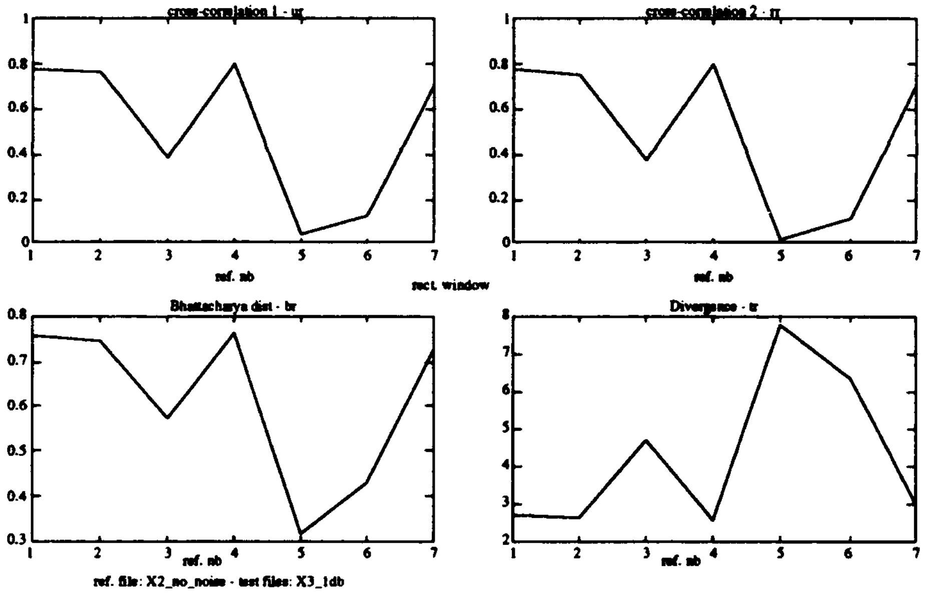
A.40: X2$_t$ correlated against X2_(-20dB)



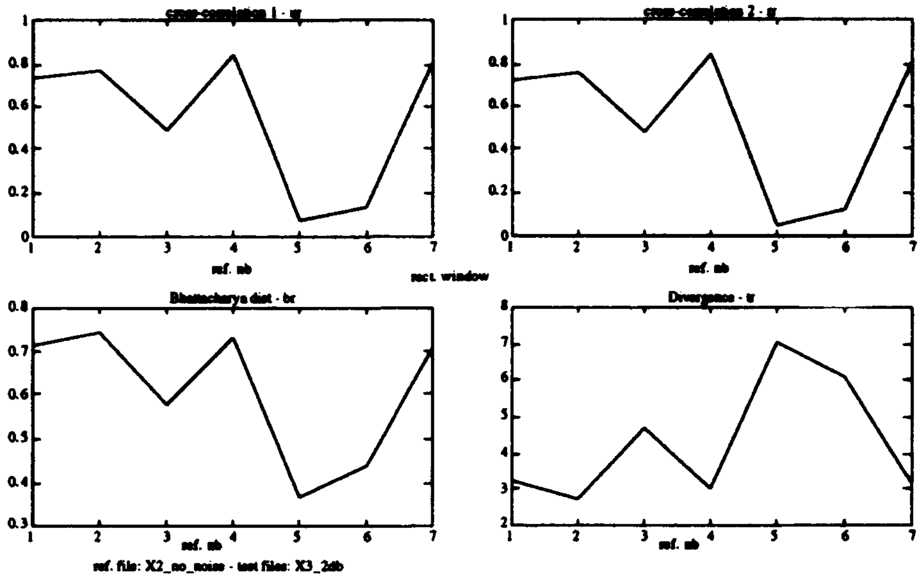
A.41: X2$_t$ correlated against X3



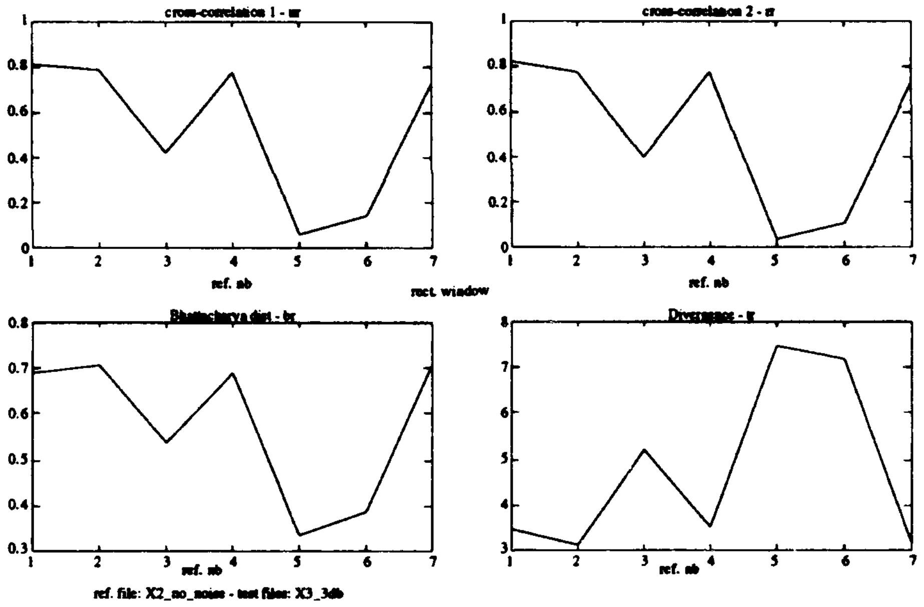
A.42: X2$_t$ correlated against X3_(0dB)



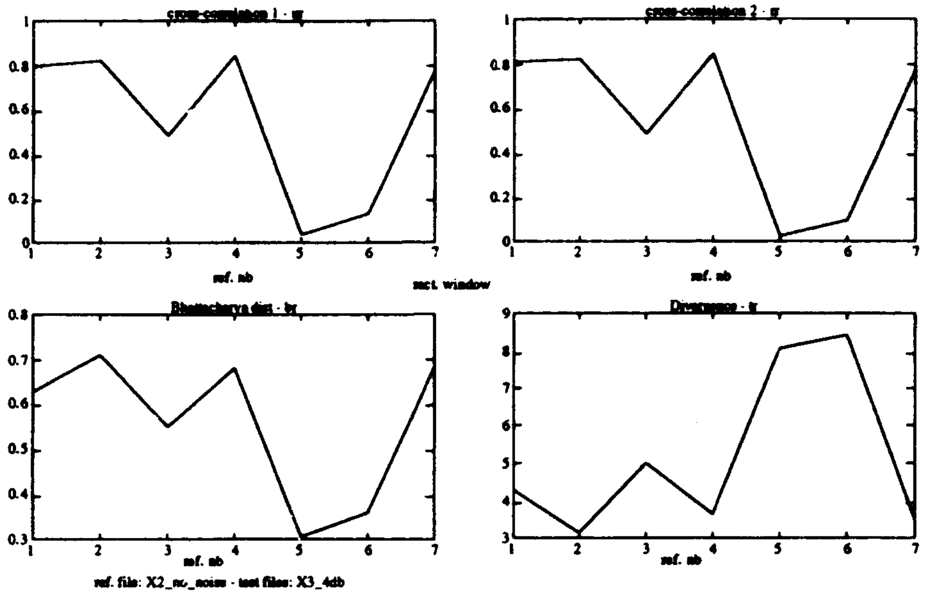
A.43: X2$_t$ correlated against X3_(-1dB)



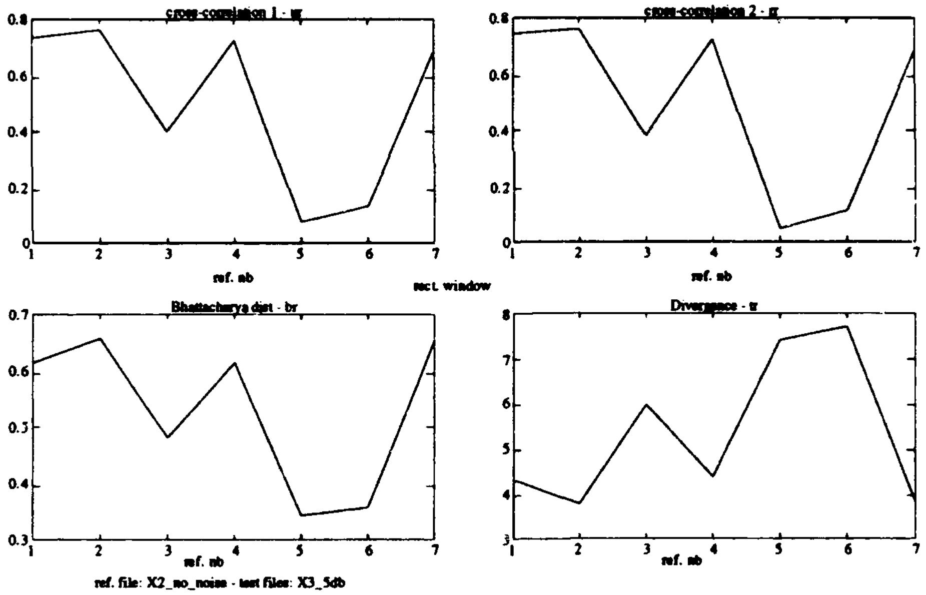
A.44: X2$_t$ correlated against X3_(-2dB)



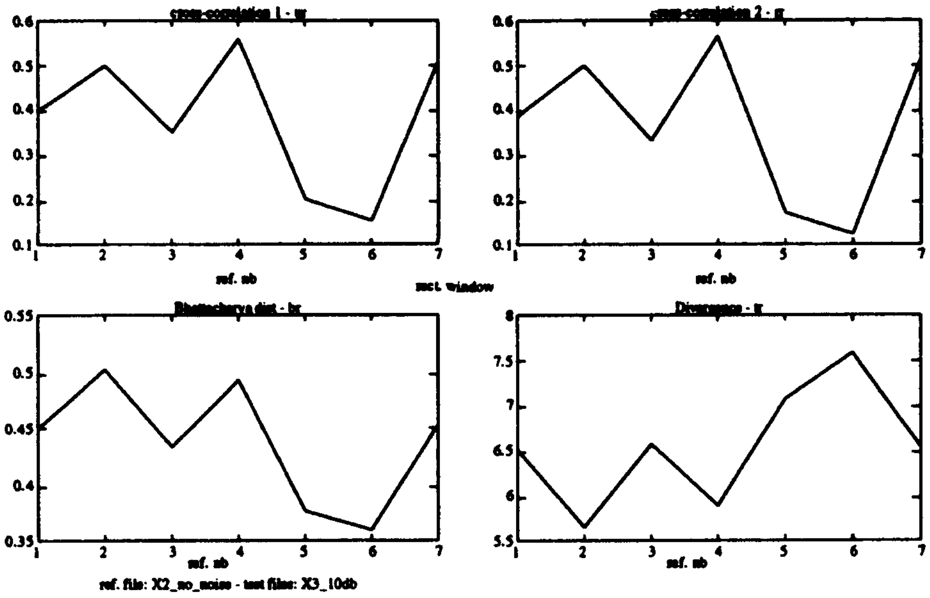
A.45: X2$_t$ correlated against X3_(-3dB)



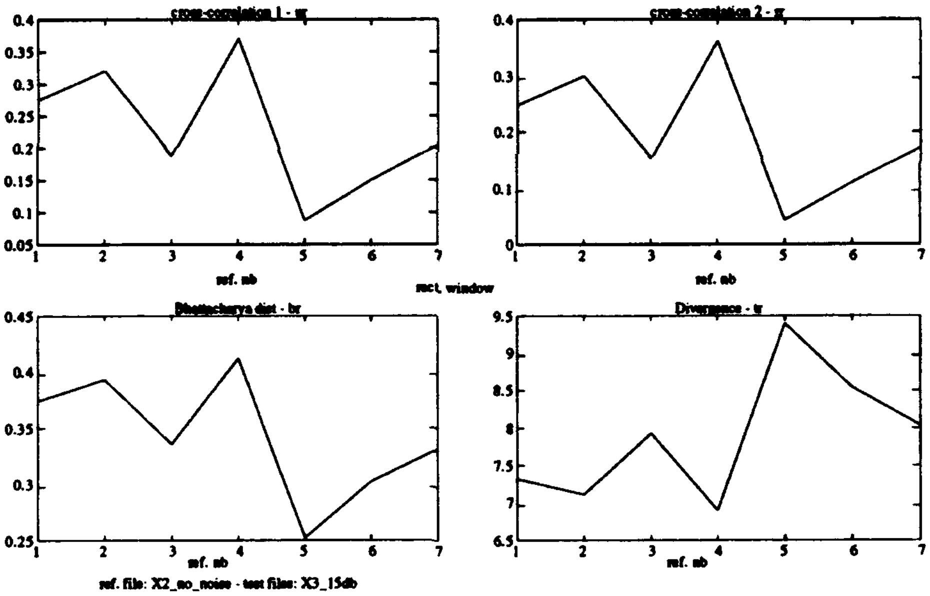
A.46: X2$_t$ correlated against X3_(-4dB)



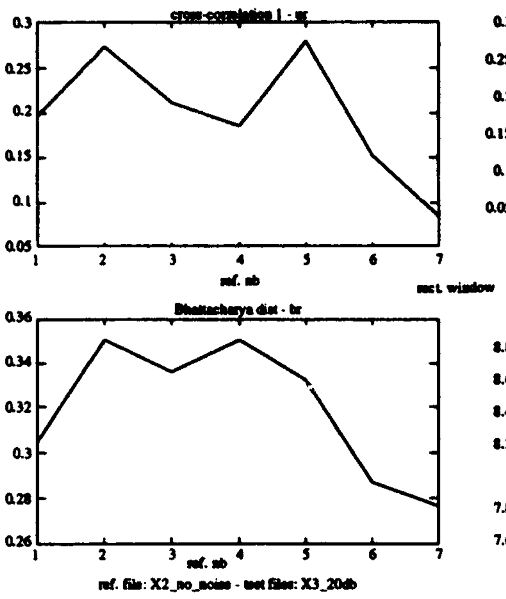
A.47: X2$_t$ correlated against X3_(-5dB)



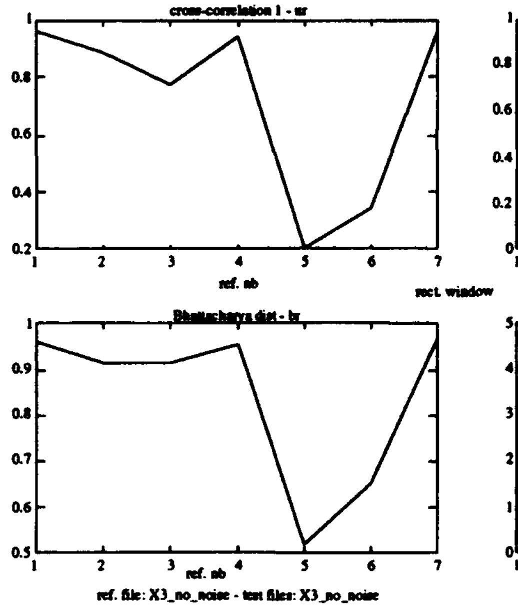
A.48: X2$_t$ correlated against X3_(-10dB)



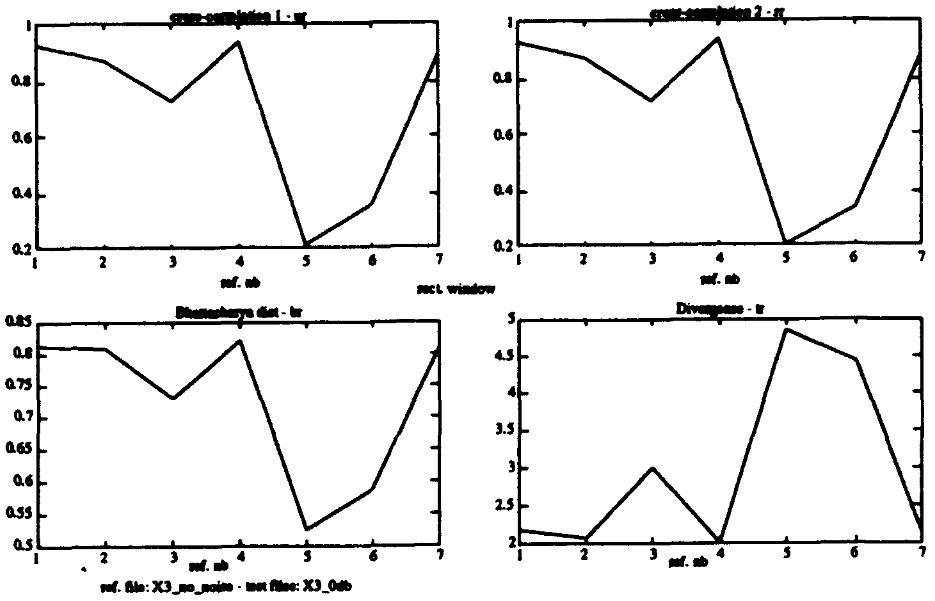
A.49: X2$_t$ correlated against X3_(-15dB)



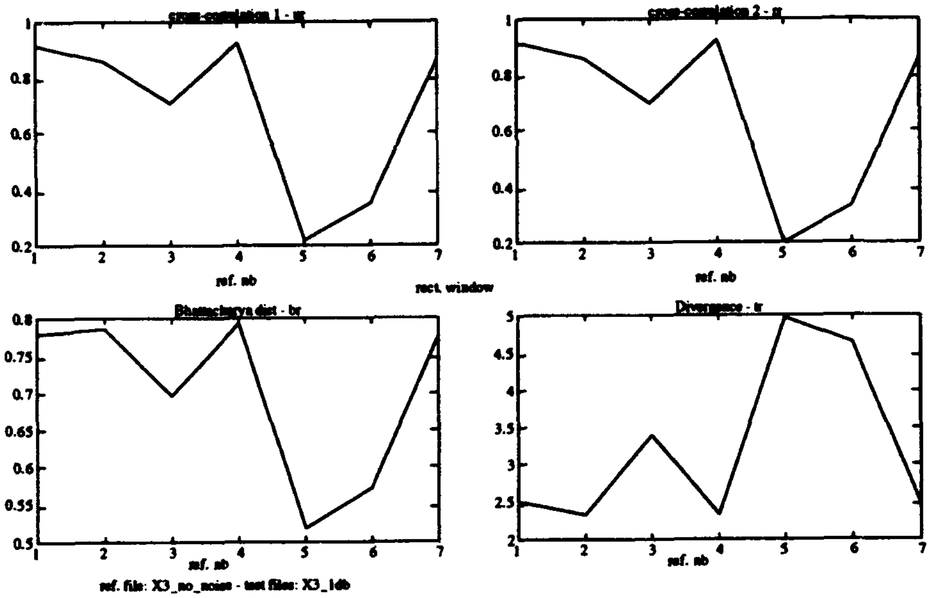
A.50: X2$_t$ correlated against X3_(-20dB)



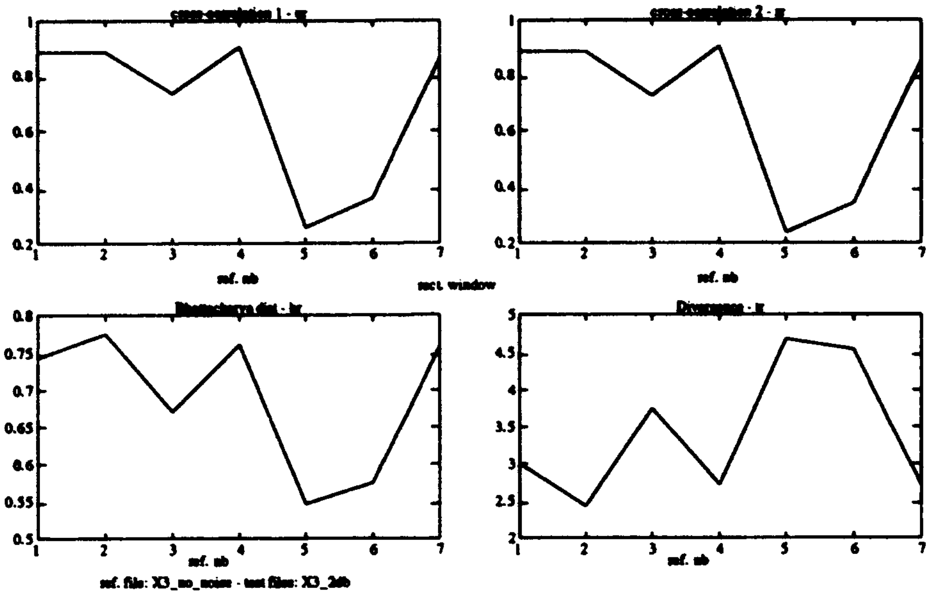
A.51: X3$_t$ correlated against X3



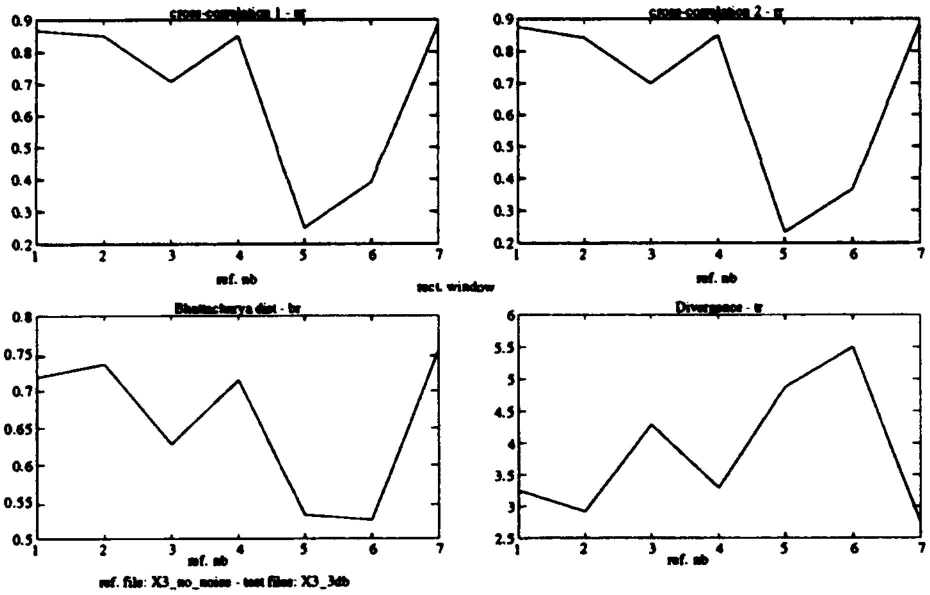
A.52: X3$_t$ correlated against X3_(0dB)



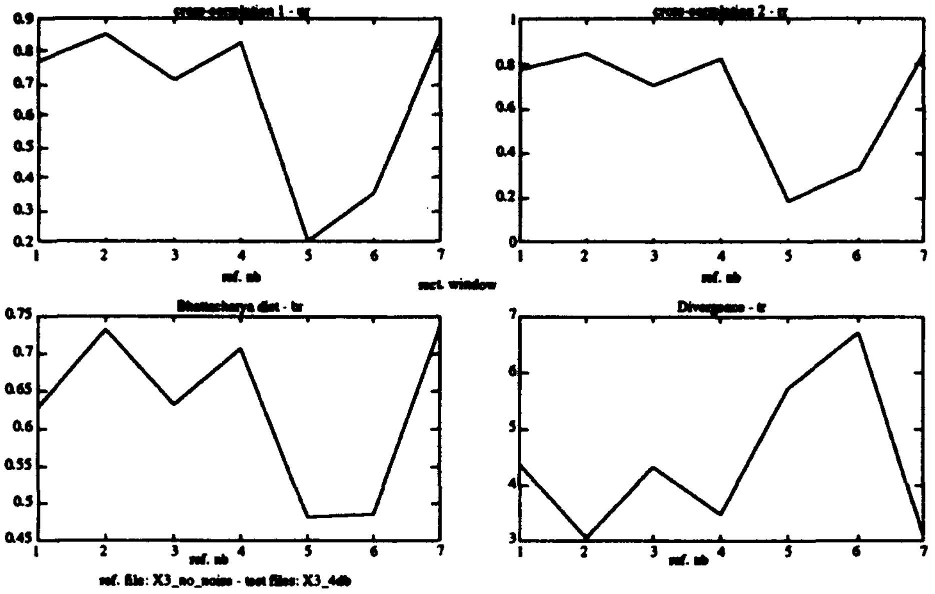
A.53: X3$_t$ correlated against X3_(-1dB)



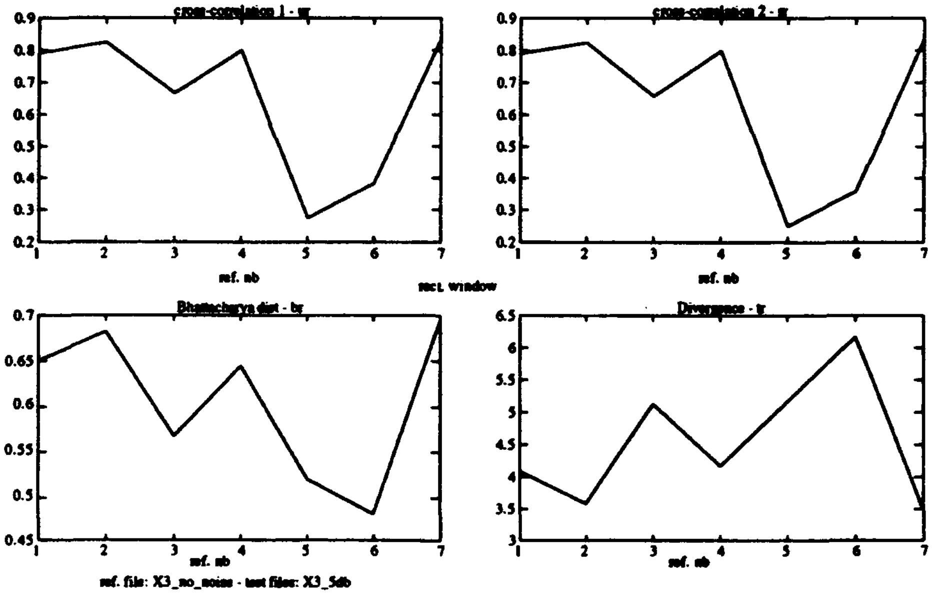
A.54: X3$_t$ correlated against X3_(-2dB)



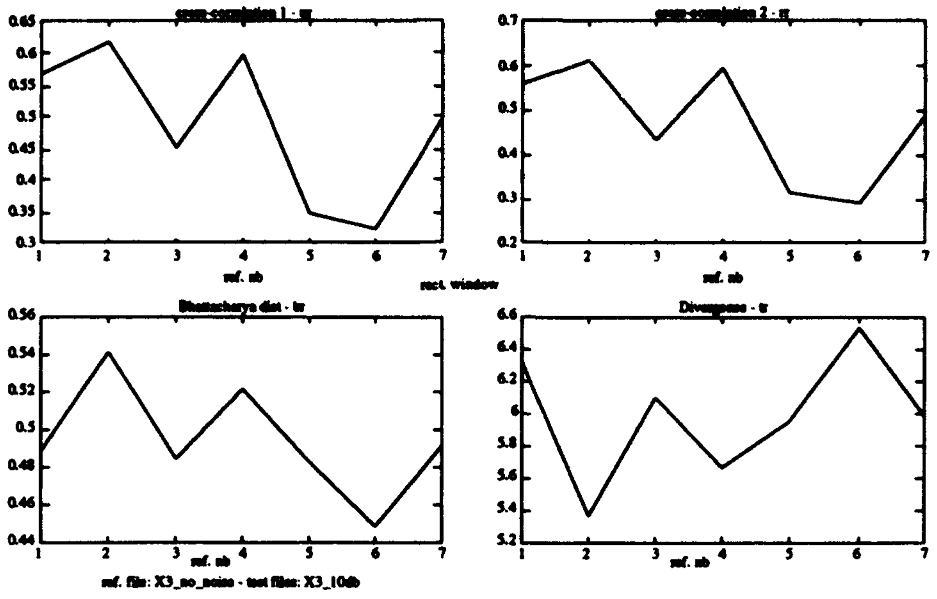
A.55: X3$_t$ correlated against X3_(-3dB)



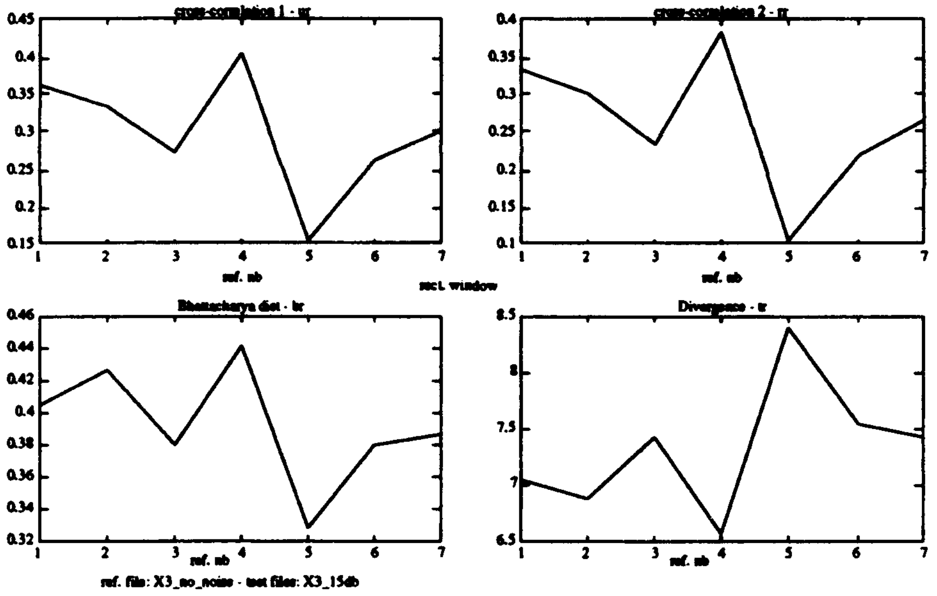
A.56: X3$_t$ correlated against X3_(-4dB)



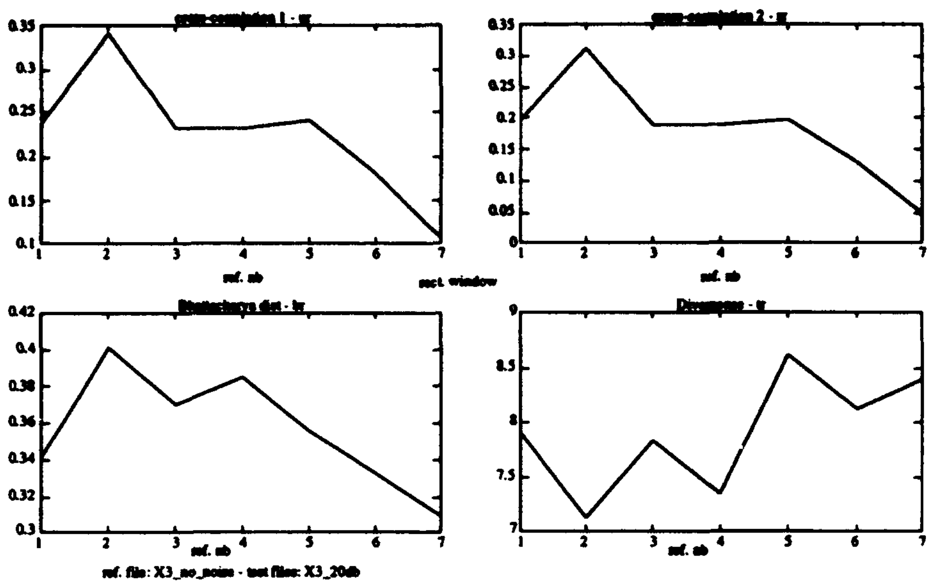
A.57: X3$_t$ correlated against X3_(-5dB)



A.58: X3$_t$ correlated against X3_(-10dB)



A.59: X3$_t$ correlated against X3_(-15dB)



A.60: X3$_t$ correlated against X3_(-20dB)

APPENDIX B. NEURAL NETWORK RESULTS

The following sections show individual and composite results for the neural network when trained and tested using $10*\log_{10}$ (dB) of each spectral coefficient. The last section presents a "snapshot" of the *NeuralWorks Professional II / Plus* network during the first 200 training cycles and after it has completed all 50,000 training cycles.

A. NEURAL NETWORK RESULTS USING dB SPECTRAL COEFFICIENTS

X1_0dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	0.985514	0.012112
1	0	1.012548	-0.010957
1	0	0.94641	0.049602
1	0	1.000553	0.003102
1	0	0.989839	0.021351
1	0	1.04876	-0.042894
1	0	1.034953	-0.034203
1	0	1.003342	-0.005121
1	0	1.011404	-0.012022
1	0	1.013739	-0.0118
1	0	1.009815	-0.000483
1	0	1.002547	0.001068
1	0	0.993314	0.014281
1	0	0.99483	0.011315
1	0	1.029343	-0.026673
1	0	0.999266	0.002761

Avg. Value :	1.004781	-0.00179
Std. Dev. :	0.023816	0.022301

B.1: X1_0DB TEST RUN USING DB COEFFICIENTS

X1_1dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	0.994937	0.007439
1	0	0.973448	0.020785
1	0	1.00657	-0.002392
1	0	1.044721	-0.037203
1	0	1.008065	-0.010653
1	0	0.972425	0.028253
1	0	1.027741	-0.023662
1	0	1.014782	-0.009005
1	0	0.991212	0.013091
1	0	0.979859	0.022586
1	0	1.045515	-0.039643
1	0	1.00014	0.003488
1	0	0.974176	0.02816
1	0	0.995495	0.003883
1	0	0.979439	0.025169
1	0	1.040204	-0.030665

Avg. Value :	1.002245	-0.000923
Std. Dev. :	0.025455	0.022205

B.2: X1_(-1DB) TEST RUN USING DB COEFFICIENTS

X1_2dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	1.002919	0.002524
1	0	0.996089	0.00675
1	0	0.986576	0.011757
1	0	0.969087	0.038583
1	0	1.024063	-0.024146
1	0	1.002859	0.002601
1	0	1.024977	-0.018437
1	0	0.992239	0.015574
1	0	0.996377	0.005607
1	0	0.973648	0.024813
1	0	0.978675	0.018617
1	0	1.013239	-0.005136
1	0	0.975667	0.028702
1	0	1.003205	-0.003464
1	0	1.009204	-0.005796
1	0	0.929035	0.076644

Avg. Value :	0.992366	0.011095
Std. Dev. :	0.023588	0.024078

B.3: X1_(-2DB) TEST RUN USING DB COEFFICIENTS

X1_3dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	0.947249	0.050366
1	0	1.013129	-0.016511
1	0	1.025961	-0.029054
1	0	0.965155	0.044801
1	0	1.020487	-0.01558
1	0	0.951926	0.051915
1	0	0.945988	0.057117
1	0	0.960217	0.044146
1	0	1.001843	0.00606
1	0	1.03419	-0.035609
1	0	0.940548	0.061293
1	0	1.007208	-0.005657
1	0	0.96777	0.036301
1	0	0.959688	0.048837
1	0	1.055456	-0.047907
1	0	1.023781	-0.024748

Avg. Value :	0.988787	0.014111
Std. Dev. :	0.037528	0.035681

B.4: X1_(-3DB) TEST RUN USING DB COEFFICIENTS

X1_4dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	1.015081	-0.01733
1	0	0.966516	0.037006
1	0	0.992448	0.010832
1	0	0.994356	0.010087
1	0	0.996341	0.007601
1	0	0.981652	0.016181
1	0	0.797317	0.202641
1	0	1.023736	-0.020674
1	0	1.040676	-0.037292
1	0	0.964215	0.049195
1	0	0.977511	0.023413
1	0	0.962923	0.039312
1	0	0.934531	0.060086
1	0	1.019561	-0.010075
1	0	1.026661	-0.018354
1	0	1.045942	-0.041317

Avg. Value :	0.983717	0.019457
Std. Dev. :	0.055592	0.057231

B.5: X1_(-4DB) TEST RUN USING DB COEFFICIENTS

X1_5dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	0.99314	0.013854
1	0	0.984991	0.026575
1	0	0.957986	0.041634
1	0	0.980571	0.027812
1	0	0.902165	0.107397
1	0	1.005979	0.002241
1	0	0.948544	0.047796
1	0	1.027172	-0.018448
1	0	0.972042	0.021708
1	0	0.951681	0.052068
1	0	0.993003	0.010887
1	0	0.942066	0.05906
1	0	0.955791	0.04488
1	0	1.020501	-0.024103
1	0	0.976964	0.026194
1	0	0.998967	0.010717

Avg. Value :	0.975723	0.026142
Std. Dev. :	0.032048	0.03171

B.6: X1_(-5DB) TEST RUN USING DB COEFFICIENTS

X1_10dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	0.883875	0.127374
1	0	1.018357	-0.01
1	0	1.026232	-0.010722
1	0	0.990116	0.009649
1	0	0.913718	0.092742
1	0	0.968729	0.036754
1	0	0.96568	0.040405
1	0	0.986839	0.015912
1	0	1.023957	-0.016056
1	0	0.94398	0.067508
1	0	0.845992	0.148149
1	0	0.922934	0.086174
1	0	0.972973	0.02241
1	0	0.933653	0.063955
1	0	0.980912	0.0193
1	0	0.983363	0.031031

Avg. Value :	0.960082	0.045287
Std. Dev. :	0.050262	0.048767

B.7: X1_(-10DB) TEST RUN USING DB COEFFICIENTS

X1_15dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	0.947945	0.062906
1	0	0.838938	0.174194
1	0	0.741815	0.274122
1	0	0.837529	0.166979
1	0	0.985457	0.012949
1	0	0.776195	0.236674
1	0	0.561357	0.461622
1	0	0.78974	0.227075
1	0	0.866889	0.136369
1	0	0.779873	0.214925
1	0	0.777662	0.219904
1	0	0.564136	0.442688
1	0	0.842061	0.141287
1	0	0.745887	0.261884
1	0	0.926287	0.087858

Avg. Value :	0.798785	0.208096
Std. Dev. :	0.120074	0.123632

B.8: X1_(-15DB) TEST RUN USING DB COEFFICIENTS

X1_20dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	0.870019	0.150788
1	0	0.711968	0.29582
1	0	0.767978	0.253558
1	0	0.913239	0.084851
1	0	0.471338	0.515487
1	0	0.841823	0.149467
1	0	0.685123	0.326985
1	0	0.902874	0.093297
1	0	0.887823	0.115002
1	0	0.745001	0.263297
1	0	0.934202	0.059282
1	0	0.909674	0.092459
1	0	0.918507	0.093795
1	0	0.924973	0.078777
1	0	0.873648	0.140561
1	0	0.829033	0.173374

Avg. Value :	0.824201	0.180425
Std. Dev. :	0.123171	0.112205

B.9: X1_(-20DB) TEST RUN USING DB COEFFICIENTS

X2_0dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.117815	0.886164
0	1	0.156994	0.837175
0	1	0.421794	0.584246
0	1	0.288242	0.714036
0	1	0.167105	0.839538

Avg. Value :	0.23039	0.772232
Std. Dev. :	0.124547	0.122937

B.10: X2_(0DB) TEST RUN USING DB COEFFICIENTS

X2_1dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.325969	0.691396
0	1	0.162992	0.830086
0	1	0.151353	0.851881
0	1	0.23785[illegible]	0.752323
0	1	0.257506	0.734005

Avg. Value :	0.227126	0.771938
Std. Dev. :	0.071873	0.067228

B.11: X2_(-1DB) TEST RUN USING DB COEFFICIENTS

X2_2dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.654015	0.358105
0	1	0.150463	0.852283
0	1	0.329286	0.682514
0	1	0.277422	0.727499
0	1	0.373423	0.621457

Avg. Value :	0.356922	0.648372
Std. Dev. :	0.185907	0.18202

B.12: X2_(-2DB) TEST RUN USING DB COEFFICIENTS

X2_3dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.298776	0.706535
0	1	0.425168	0.590307
0	1	0.086257	0.921049
0	1	0.173117	0.833161
0	1	0.464906	0.55448

Avg. Value :	[illegible]	[illegible]
Std. Dev. :	[illegible]	[illegible]

B.13: X2_(-3DB) TEST RUN USING DB COEFFICIENTS

X2_4dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.173572	0.834815
0	1	0.256011	0.740416
0	1	0.355436	0.651025
0	1	0.447913	0.545617
0	1	0.316467	0.679035

Avg. Value :	0.30988	0.690182
Std. Dev. :	0.103284	0.107237

B.14: X2_(-4DB) TEST RUN USING DB COEFFICIENTS

X2_5dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.23532	0.778129
0	1	0.248206	0.750971
0	1	0.0736	0.931462
0	1	0.674684	0.328779
0	1	0.472088	0.530325

Avg. Value :	0.34078	0.663933
Std. Dev. :	0.23442	0.235779

B.15: X2_(-5DB) TEST RUN USING DB COEFFICIENTS

X2_10dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.854568	0.164976
0	1	0.373137	0.615863
0	1	0.529754	0.468111
0	1	0.626614	0.371361
0	1	0.694111	0.322639

Avg. Value :	0.615637	0.38859
Std. Dev. :	0.179960	0.167801

B.16: X2_(-10DB) TEST RUN USING DB COEFFICIENTS

X2_15dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.797481	0.215375
0	1	0.69267	0.305484
0	1	0.525645	0.472482
0	1	0.639251	0.359473
0	1	0.738095	0.264934

Avg. Value :	0.678628	0.32355
Std. Dev. :	0.103458	0.09866

B.17: X2_(-15DB) TEST RUN USING DB COEFFICIENTS

X2_20dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.475234	0.53427
0	1	0.711444	0.296867
0	1	0.891632	0.125975
0	1	0.658254	0.361278
0	1	0.801026	0.220636

Avg. Value :	0.707518	0.307805
Std. Dev. :	0.157353	0.15405

B.18: X2_(-20DB) TEST RUN USING DB COEFFICIENTS

X1

	1		0	
Tested Sets	Avg.	Std. Dev.	Avg.	Std. Dev.
X1_(0dB)	1.00476106	0.02301461	-0.0017851	0.0223812
X1_(-1dB)	1.00304556	0.02545542	-0.0000231	0.02320481
X1_(-2dB)	0.99236619	0.02388753	0.01094956	0.02407846
X1_(-3db)	0.98878725	0.03752863	0.01411063	0.03858143
X1_(-4dB)	0.98371669	0.05859173	0.019457	0.05733079
X1_(-5dB)	0.97572269	0.03204829	0.028142	0.03170962
X1_(-10dB)	0.96008188	0.05035168	0.04528656	0.04876679
X1_(-15dB)	0.79878473	0.12007384	0.20809573	0.12363316
X1_(-20dB)	0.82420144	0.12312122	0.180425	0.12220495

X2

	0		1	
Tested Sets	Avg.	Std. Dev.	Avg.	Std. Dev.
X2_(0dB)	0.23039	0.12454662	0.7722318	0.12293667
X2_(-1dB)	0.2271358	0.0718728	0.7719382	0.06723605
X2_(-2dB)	0.3569218	0.18590713	0.6483716	0.18302036
X2_(-3db)	0.2896448	0.16133175	0.7211064	0.15621975
X2_(-4db)	0.3098798	0.10326372	0.6901816	0.10723657
X2_(-5dB)	0.3407796	0.23441981	0.6639332	0.23577924
X2_(-10dB)	0.6156368	0.17996446	0.38859	0.16779914
X2_(-15dB)	0.6786284	0.10345758	0.3235496	0.09866008
X2_(-20dB)	0.707518	0.15735304	0.3078052	0.1540503

B.19: AVERAGE FOR ALL RUNS USING DB COEFFICIENTS

B. NEURAL NETWORK RESULTS USING SPECTRAL COEFFICIENTS

X1_0dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	1.02491	-0.021512
1	0	1.013492	-0.003956
1	0	1.004652	-0.025819
1	0	1.016763	0.002389
1	0	1.037981	-0.013527
1	0	1.035746	-0.015367
1	0	1.043003	-0.033754
1	0	1.027983	-0.037629
1	0	1.021059	-0.010246
1	0	1.039124	-0.014804
1	0	1.015891	-0.016927
1	0	1.01275	-0.006096
1	0	0.98715	0.041793
1	0	0.955978	0.105051
1	0	0.991863	0.010704
1	0	1.051338	-0.033154

Avg. Value :	[illegible]	[illegible]
Std. Dev. :	[illegible]	[illegible]

B.20: X1_(0DB) TEST RUN

X1_1dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	1.021395	-0.013044
1	0	1.029132	-0.042669
1	0	1.037913	-0.023819
1	0	1.026461	-0.027834
1	0	0.996595	-0.03155
1	0	1.02973	-0.025064
1	0	1.039016	-0.019605
1	0	1.023483	-0.011526
1	0	1.038391	-0.0239
1	0	1.011946	-0.014063
1	0	1.037825	-0.023611
1	0	1.037025	-0.015577
1	0	1.041583	-0.035297
1	0	1.018065	-0.015267
1	0	1.034498	-0.034541
1	0	1.038844	-0.013959

Avg. Value :	[illegible]	[illegible]
Std. Dev. :	[illegible]	[illegible]

B.21: X1_(-1DB) TEST RUN

X1_2dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	1.025311	-0.006925
1	0	1.021909	-0.003156
1	0	1.024667	-0.017509
1	0	0.735648	0.196919
1	0	1.013083	-0.018537
1	0	1.03055	-0.011811
1	0	1.041829	-0.019335
1	0	1.042587	-0.015612
1	0	1.028234	-0.013741
1	0	1.01007	-0.025618
1	0	0.97209	0.023154
1	0	1.032259	-0.01667
1	0	1.002706	0.005924
1	0	0.925592	0.089042
1	0	1.033942	-0.032051
1	0	1.024594	-0.023438

Avg. Value :	[illegible]	[illegible]
Std. Dev. :	[illegible]	[illegible]

B.22: X1_(-2DB) TEST RUN

X1_3dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	1.007387	-0.00325
1	0	1.026007	-0.021023
1	0	1.015069	-0.01574
1	0	1.023958	-0.009288
1	0	1.040122	-0.036837
1	0	1.00329	-0.011326
1	0	0.989575	0.021835
1	0	0.968216	0.048321
1	0	1.031374	-0.010962
1	0	1.016075	-0.014745
1	0	1.028289	-0.004799
1	0	1.044034	-0.017687
1	0	0.989745	0.029389
1	0	1.037074	-0.01035
1	0	1.039801	-0.016369
1	0	1.016384	-0.009391
	Avg. Value :	1.017276	-0.008314
	Std. Dev. :	0.021203	0.021072

B.23: X1_(-3DB) TEST RUN

X1_4dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	0.99691	0.014118
1	0	1.029904	-0.01223
1	0	1.044623	-0.023896
1	0	1.020535	-0.022418
1	0	1.009989	0.004466
1	0	1.043172	-0.015444
1	0	0.769076	0.192055
1	0	1.062547	-0.032375
1	0	1.041744	-0.016585
1	0	1.01051	-0.000607
1	0	0.967084	0.043551
1	0	0.932738	0.035637
1	0	1.010362	-0.045238
1	0	1.003831	0.001989
1	0	1.043786	-0.016485
1	0	1.044384	-0.021139
	Avg. Value :	1.00198	0.005327
	Std. Dev. :	0.070203	0.054897

B.24: X1_(-4DB) TEST RUN

X1_5dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	1.042961	-0.015735
1	0	1.010399	0.002472
1	0	1.035681	-0.02502
1	0	0.932735	0.064822
1	0	1.039351	-0.014759
1	0	1.052467	-0.024027
1	0	0.937426	0.070784
1	0	1.042331	-0.015593
1	0	1.013376	-0.007081
1	0	0.986303	0.008698
1	0	1.038716	-0.015407
1	0	0.854373	0.138406
1	0	1.015919	-0.0121
1	0	1.032529	-0.063556
1	0	0.702426	0.299126
1	0	1.00384	0.046133

Avg. Value :	[illegible]	[illegible]
Std. Dev. :	[illegible]	[illegible]

B.25: X1_(-5DB) TEST RUN

X1_10dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	1.007762	0.016115
1	0	1.04475	-0.017211
1	0	1.04352	-0.017045
1	0	1.044941	-0.017361
1	0	1.019833	0.004041
1	0	0.975041	0.094982
1	0	1.044635	-0.017168
1	0	1.042094	-0.01669
1	0	1.037353	-0.012267
1	0	0.907289	0.075925
1	0	1.042963	-0.015612
1	0	0.741976	0.304724
1	0	1.021775	0.001911
1	0	0.809704	0.176132
1	0	1.030621	-0.025276
1	0	1.042517	-0.018122

Avg. Value :	0.991048	0.032317
Std. Dev. :	0.09212	0.091368

B.26: X1_(-10DB) TEST RUN

X1_15dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	1.044568	-0.017123
1	0	1.043215	-0.016486
1	0	0.879885	0.123394
1	0	1.033399	-0.028427
1	0	0.844725	0.154753
1	0	1.044736	-0.017204
1	0	0.981499	0.038596
1	0	0.72067	0.334055
1	0	1.022012	-0.006206
1	0	1.044073	-0.01643
1	0	1.044671	-0.017187
1	0	1.032771	-0.029025
1	0	0.687611	0.246589
1	0	1.029817	0.007382
1	0	1.037477	-0.013628
1	0	0.989206	-0.039103

Avg. Value :	0.967521	0.043917
Std. Dev. :	0.119821	0.111647

B.27: X1_(-15DB) TEST RUN

X1_20dB

NN ideal output		NN tested output	
X1	X2	X1	X2
1	0	0.975478	0.092157
1	0	0.820394	0.177842
1	0	0.702694	0.27107
1	0	1.042918	-0.01418
1	0	1.041825	-0.014615
1	0	0.933669	0.032133
1	0	0.618574	0.241181
1	0	0.765128	0.207375
1	0	1.044734	-0.017203
1	0	1.037327	-0.004915
1	0	0.706115	0.178429
1	0	1.044248	-0.017002
1	0	1.044749	-0.017211
1	0	0.824866	0.173036
1	0	0.585622	0.39251
1	0	0.758306	0.227208

Avg. Value :	0.871665	0.119238
Std. Dev. :	0.16751	0.130921

B.28: X1_(-20DB) TEST RUN

X2_0dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.418781	0.573938
0	1	0.594673	0.315906
0	1	0.471135	0.46894
0	1	0.538329	0.415411
0	1	0.735596	0.28973
	Avg. Value :	[illegible]	[illegible]
	Std. Dev. :	[illegible]	[illegible]

B.29: X1_(0DB) TEST RUN

X2_1dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.922756	0.061662
0	1	0.223926	0.71707
0	1	0.981132	0.018058
0	1	0.709805	0.337266
0	1	0.838353	0.147331
	Avg. Value :	[illegible]	[illegible]
	Std. Dev. :	[illegible]	[illegible]

B.30: X2_(-1DB) TEST RUN

X2_2dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.804211	0.177829
0	1	0.497565	0.399119
0	1	0.715526	0.276566
0	1	0.637539	0.339258
0	1	0.755016	0.259407
	Avg. Value :	0.681971	0.290436
	Std. Dev. :	0.119775	0.083712

B.31: X2_(-2DB) TEST RUN

X2_3dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	1.0241	-0.00419
0	1	0.953266	0.051623
0	1	0.480629	0.507723
0	1	0.128375	0.871549
0	1	1.037908	-0.011096

Avg. Value :	0.724856	0.283122
Std. Dev. :	[illegible]	[illegible]

B.32: X2_(-3DB) TEST RUN

X2_4dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	1.024049	-0.003291
0	1	0.249717	0.705029
0	1	0.509559	0.493395
0	1	0.995888	0.022901
0	1	0.652073	0.279706

Avg. Value :	0.686257	0.299548
Std. Dev. :	[illegible]	0.304397

B.33: X2_(-4DB) TEST RUN

X2_5dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.738745	0.211714
0	1	1.002396	0.003658
0	1	0.783052	0.221736
0	1	1.035853	-0.014408
0	1	1.016424	-0.007589

Avg. Value :	0.915294	0.083022
Std. Dev. :	0.142308	0.122275

B.34: X2_(-5DB) TEST RUN

X2_10dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	1.044749	-0.017211
0	1	0.903357	0.104764
0	1	0.824273	0.172092
0	1	1.038665	-0.011961
0	1	0.869158	0.108793

Avg. Value :	0.936040	0.071295
Std. Dev. :	[illegible]	[illegible]

B.35: X2_(-10DB) TEST RUN

X2_15dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	1.044748	-0.01721
0	1	0.721217	0.33342
0	1	0.930551	0.082492
0	1	0.918648	0.110256
0	1	1.044747	-0.017215

Avg. Value :	0.931982	0.098349
Std. Dev. :	[illegible]	[illegible]

B.36: X2_(-15DB) TEST RUN

X2_20dB

NN ideal output		NN tested output	
X1	X2	X1	X2
0	1	0.721196	0.333437
0	1	1.044528	-0.017184
0	1	1.043985	-0.016838
0	1	1.041209	-0.018027
0	1	0.743722	0.223446

Avg. Value :	0.918928	0.100967
Std. Dev. :	0.170413	0.168814

B.37: X2_(-20DB) TEST RUN

X1

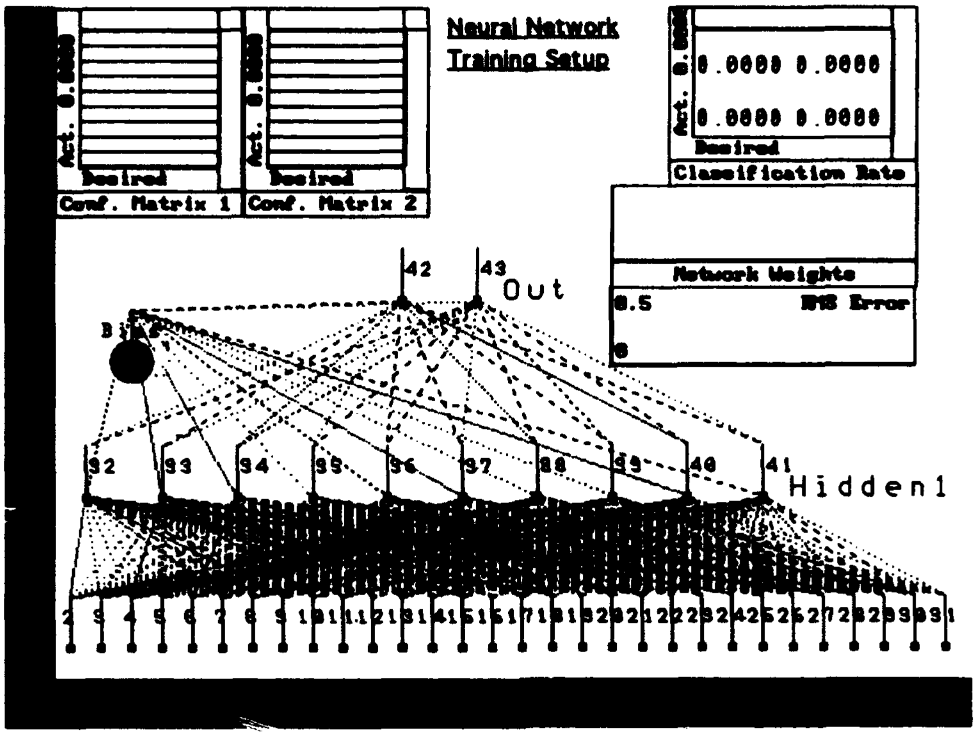
	1		0	
Tested Sets	Avg.	Std. Dev.	Avg.	Std. Dev.
X1_(0dB)	1.01748019	0.02417026	-0.0045534	0.03505935
X1_(-1dB)	1.02886888	0.01225069	-0.0232079	0.00928075
X1_(-2dB)	0.99781694	0.07581627	0.00691475	0.0581243
X1_(-3db)	1.017275	0.02130865	-0.0051389	0.02107345
X1_(-4dB)	1.00194969	0.07020321	0.00533744	0.0549967
X1_(-5dB)	0.98380206	0.09183735	0.02732269	0.08736725
X1_(-10dB)	0.99104838	0.09213024	0.03231738	0.09130805
X1_(-15dB)	0.96752094	0.11902136	0.04399688	0.1116471
X1_(-20dB)	0.87166544	0.16751046	0.11923844	0.13092134

X2

	0		1	
Tested Sets	Avg.	Std. Dev.	Avg.	Std. Dev.
X2_(0dB)	0.5517028	0.1224759	0.412785	0.11582168
X2_(-1dB)	0.7351944	0.30345772	0.2562774	0.28521129
X2_(-2dB)	0.6819714	0.11977469	0.2904358	0.08371245
X2_(-3db)	0.7248556	0.40470856	0.2831218	0.39350862
X2_(-4db)	0.6862572	0.3289881	0.299548	0.30439704
X2_(-5dB)	0.915294	0.14230777	0.0830222	0.12227511
X2_(-10dB)	0.9360404	0.10047734	0.0712954	0.08284199
X2_(-15dB)	0.9319822	0.13231954	0.0983486	0.14349261
X2_(-20dB)	0.918928	0.17041299	0.1009668	0.16661382

B.38: AVERAGE FOR ALL RUNS USING SPECTRAL COEFFICIENTS

C. "SNAPSHOT" OF NEURAL NETWORK



B.39: Neural Network Training Setup

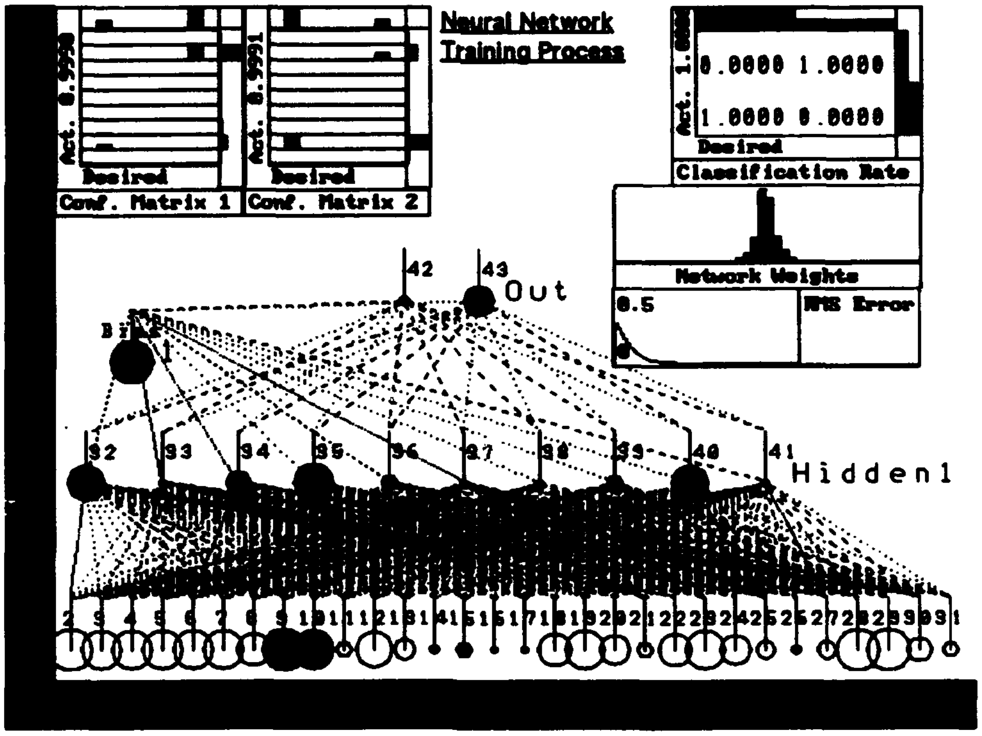
The tools used in the NN analysis are displayed in Figures B.39 through B.42. The small squares and circles at the bottom and in between each NN configuration represent initial weights to be applied to each of the 30 input values. As the training process continues these weighs or processing elements, (PE) will grow and shrink into different sizes. The larger the circle, the larger the weight for that PE.

The Confusion Matrix tool gives a visual indication of how well the network is doing. Ideally what one would like to have the bins (cells in each matrix) on the diagonal from the lower left to the upper right showing the highest counts. This indicates the network is giving an output that matches the desired output [8]

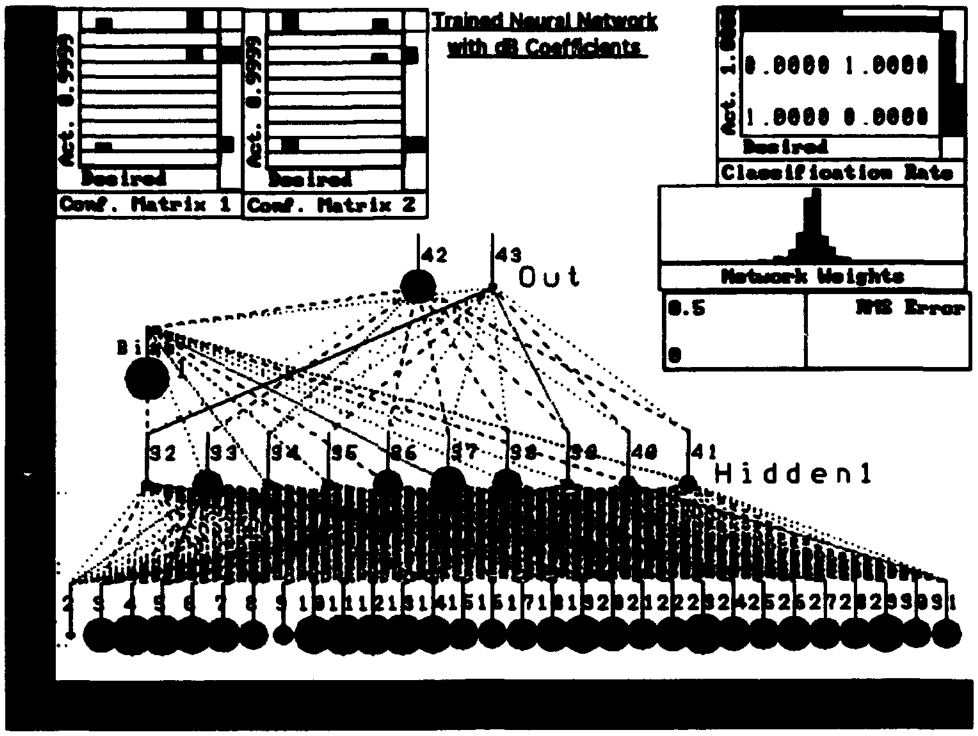
The histogram at the top of the Classification Rate tool represents the desired results. The histogram of the rows represents the actual network predictions. The body of the Classification Rate tool presents the intersection of the desired results and actual network predictions [8].

The Network Weights tool shows the weight distributions as they are updated in the training process. Initially, the weights will start out close to zero. As the network is trained, some weights will move away from their near zero starting points. [8]

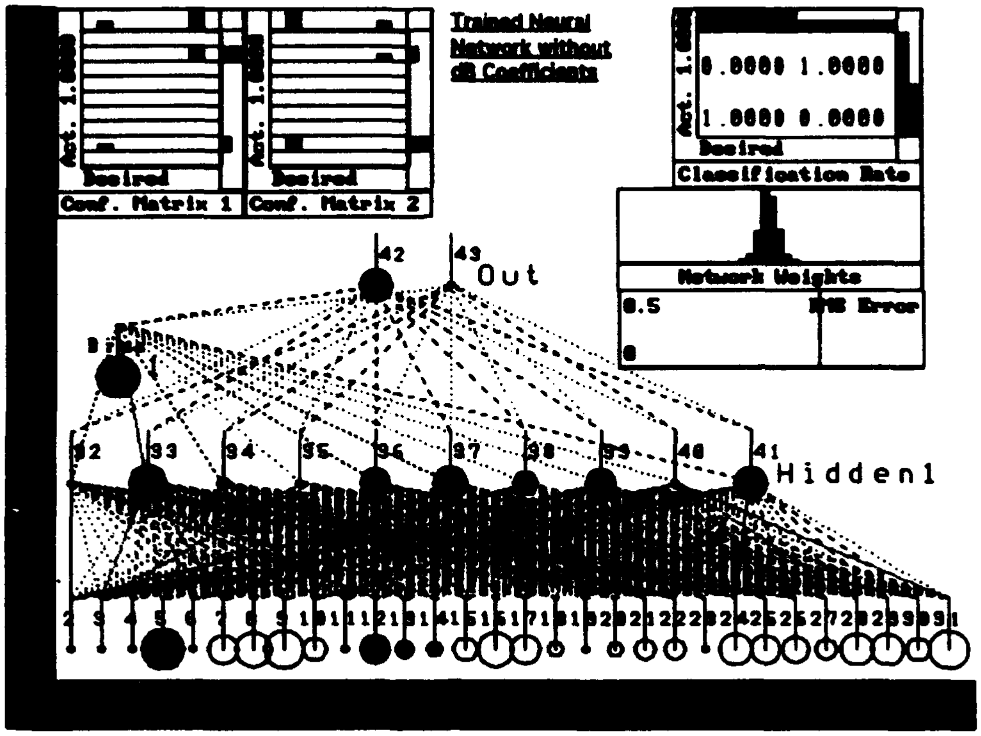
The RMS error tool measures the root mean square error for all processing elements in the output layer as the NN is being trained. [8]



B.40: Neural Network Training in Process



B.41: Trained Neural Network using dB Coefficients



B.42: Trained Neural Network using Spectral Coefficients

APPENDIX C. MATLAB® CODE IMPLEMENTATION

The following programs were written for the spectral measure techniques discussed in Chapters II and III of the text and for the training and testing sets used with the NN in Chapter IV.

A. SINGLE RUN PROGRAMS

C.1: Program no_noise LOFAR.M [1]

```
% This program computes the LOFARGRAM based on windowed periodograms and display
%(modified from the original program completed by Dr. Ralph Hippenstiel and Dr.
%Monique P. Fargues)
clear pow
count=input('How many no_noise input files would you like to enter » ');
for loopct = 1:count
fname=input('entire name of input file is (w/o ext) » ','s');
eval(['load ', fname,'.dat']);
name=eval(fname);
eval(['clear ',fname]);
iopt=input('mean removed: y/n=1/0 » ');
int=input('data truncation for exc210-220-230: y/n=1/0 » ');
isq=input('get power spectrum (1) or abs value (0)» ');
iwin=input('use hamming window (1) or not (0)» ');
[drow,dcolumn]=size(name);
if drow~=1
        name=name.';
end
len=length(name);
id=input(' No. of non-zero data points in transform        ');
step=input(' shift (step size) in data points      ');
tlen=input('transform length        ');
i=fix((len-id)/step)+1;
if int==1  % data truncation for exc210-220-230
  id2=408;
end
if iwin==1,win=hamming(id).';
elseif iwin==0, win=ones(1,id); end
for ic=1:i
clear temp temp0 temp2
% interpolate data for exca210
 if int==1
```

```
    temp0(1:id2)=name(1+(ic-1)*step:id2+(ic-1)*step);
   temp2=interp(temp0,5);
   temp=decimate(temp2,2,30,'fir1');
 else
    temp(1:id)=name(1+(ic-1)*step:id+(ic-1)*step);
 end

if iopt==1      % for data mean removed
  temp=temp-mean(temp);
end
                 % direct comput. for data mean kept
%ppow=abs(fft(name(1+(ic-1)*step:id+(ic-1)*step).*win,tlen));
ppow=abs(fft(temp.*win,tlen));  % for sqrt(power spectrum) eval.
if isq==1, ppow=ppow.^2; end     % for power spectrum eval.
pow(ic,:)=ppow(1:tlen/2);
end
powc=flipud(pow);
clear name drow dcolumn len id step tlen i ic win ppow
%*************************************************************
% THIS PORTION OF THE PROGRAM ASSIGNS UNIQUE VARIABLE NAME
% TO EACH INPUT
%*************************************************************
if      fname=='EXCA200'
        pow_200=pow;
        powc_200=powc;

elseif   fname=='EXCA210'
        pow_210=pow;
        powc_210=powc;

elseif   fname=='EXCA220'
        pow_220=pow;
        powc_220=powc;

elseif   fname=='EXCA230'
        pow_230=pow;
        powc_230=powc;
end
end
X1_no_noise=X1_fcn(pow_200,pow_210,pow_220,pow_230);
X2_no_noise=X2_fcn(pow_220);
X3_no_noise=X3_fcn(pow_210,pow_220,pow_230);
fprintf('print contours using powc -plot using pow transpose \n')
```

C.2: Program noisy_LOFAR.M [1]

```
% This program computes the LOFARGRAM based on windowed periodograms and displays
%the results in a waterfall type fashion
%(modified from the original program completed by Dr. Ralph Hippenstiel and Dr.
%Monique P. Fargues)

clear pow
% » = " shift, option, \"
%    ****************************************
% THIS PORTION OF THE PROGRAM CORUPTS
% ALL THE SIGANLS WITH NOISE
%   ****************************************
count=input('How many noisy_input files would you like to enter » ');
for loopcount=1:count;
fname=input(' number of each noisy signal is » ','s');
nname=input(' what negative decibel do you want » ','s');
scale=eval(nname);
eval(['load EXCA',fname,'.dat']);
RAWDATA=eval(['EXCA',fname]);
% the s means input willbe string variable
eval(['noisy_sig_',nname,'db_',fname,'=noisy_sig(RAWDATA,scale);']);
name=eval(['noisy_sig_',nname,'db_',fname]);
eval(['clear ',fname]);
% now we clear orig. data
name=eval(['noisy_sig_',nname,'db_',fname]);
iopt=input('mean removed: y/n=1/0 » ');
int=input('data truncation for exc210-220-230: y/n=1/0 » ');
isq=input('get power spectrum (1) or abs value (0)» ');
iwin=input('use hamming window (1) or not (0)» ');
[drow,dcolumn]=size(name);
if drow~=1
        name=name.';
end
len=length(name);
id=input(' No. of non-zero data points in transform        ');
step=input(' shift (step size) in data points      ');
tlen=input('transform length        ');
i=fix((len-id)/step)+1;
if int==1  % data truncation for exc210-220-230
  id2=408;
end
if iwin==1,win=hamming(id).';
elseif iwin==0, win=ones(1,id); end
for ic=1:i
clear temp temp0 temp2
% interpolate data for exca210
 if int==1
    temp0(1:id2)=name(1+(ic-1)*step:id2+(ic-1)*step);
   temp2=interp(temp0,5);
   temp=decimate(temp2,2,30,'fir1');
 else
    temp(1:id)=name(1+(ic-1)*step:id+(ic-1)*step);
 end

if iopt==1       % for data mean removed
  temp=temp-mean(temp);
end
             % direct comput. for data mean kept
%ppow=abs(fft(name(1+(ic-1)*step:id+(ic-1)*step).*win,tlen));
ppow=abs(fft(temp.*win,tlen));   % for sqrt(power spectrum) eval.
if isq==1, ppow=ppow.^2; end      % for power spectrum eval.
pow(ic,:)=ppow(1:tlen/2);
end
powc=flipud(pow);
```

```
clear name drow dcolumn len id step tlen i ic win ppow
eval(['EXCA',fname,'_',nname,'db_pow=pow;']);
eval(['EXCA',fname,'_',nname,'db_powc=powc;']);
end
eval(['X1_',nname,'db=X1_fcn(EXCA200_',nname,'db_pow,EXCA210_',nname,'db_pow,EXC
A220_',nname,'db_pow,EXCA230_',nname,'db_pow);']);
eval(['X2_',nname,'db=X2_fcn(EXCA220_',nname,'db_pow);']);
eval(['X3_',nname,'db=X3_fcn(EXCA210_',nname,'db_pow,EXCA220_',nname,'db_pow,EXC
A230_',nname,'db_pow);']);
fprintf('print contours using powc -plot using pow transpose \n')
```

C.3: Program DIST.M [1]

```
clear ur y rr br tr mr x ref refn refd xeps yeps
%*****************************************************************
% THIS IS THE UPDATED, AUTOMATED "DIST" FUNCTION.
%*****************************************************************
%(modified from the original program completed by Dr. Ralph Hippenstiel and Dr.
%Monique P. Fargues)
% this function computes the different distances
% rr is the normalized cross correlation (no DC component present)
% ur is the normalized cross correlation (includes potential DC effects)
% tr is the Divergence (Kullman-Leibner)
% br is the Bhattacharyya distance (coefficient implemented)
% mr is the Matsushita distance
% xfile is the input array 1st idex is pulse Number
%                   2nd idex is freq. (i.e. bin number)
% function [ur,rr,br,tr,mr] = dist(reference,xfile)
% ref is the templated (reference signal)
% function [ur,rr,br,tr,mr]=dist(ref,x)
%*****************************************************************
RUNS=input('How many times do you which to run this program ?? » ');
for Run=1:RUNS
clear ur y rr br tr mr x ref refn refd xeps yeps
file_ref=input('input ref.file X1,X2 or X3 » ','s');
ref0=eval([file_ref,'_no_noise']);
ref0=log(ref0); % THIS TAKES THE LOG ... GET RID OF
file_test=input('input test files » ','s');
x=eval([file_test]);
x=log(x);      % THIS TAKES THE LOG ... GET RID OF
%iplot=input('plot y/n=1/0: ');
iplot=1;
if iplot==1
%iwin=input('data used rect wind. (0) or hamming wind. (1): ');
iwin=0;
test_fil=file_test;
isq=0;
end
if isq==1 % square input data for run in bad3 and good3 (no square
  ref0=ref0.^2;      % factor present in lofar.m up to 2/9/93
   x=x.^2;
end
```

```
% compute reference template
ref=mean(ref0);
[n,b]=size(x);
refn=ref-mean(ref);
refd=ref/sum(ref);  %sum over each col. to get norm. ref.
refn=refn+eps*ones(refn);refd=refd+eps*ones(refd);
for i=1:n
xeps(i,:)=x(i,:)+eps*ones(x(i,:));
ur(i)=ref*x(i,:)'/(sqrt(ref*ref'*xeps(i,:)*xeps(i,:)'));
y(i,:)=x(i,:)-mean(x(i,:));
yeps(i,:)=y(i,:)+eps*ones(y(i,:));
rr(i)=refn*y(i,:)'/(sqrt(refn*refn'*yeps(i,:)*yeps(i,:)'));
  x(i,:)=xeps(i,:)/sum(xeps(i,:)); % normalization over row for pdf
br(i)=sum(sqrt(refd.*x(i,:)));
tr(i)=sum((log(refd./x(i,:)).*refd)-(log(refd./x(i,:)).*x(i,:)));
mr(i)=sqrt( sum(( sqrt(x(i,:)) - sqrt(refd) ).^2) );
%pause
end
%ref_f=num2str(file_ref);
test_f=num2str(test_fil);
v=['ref. file: ',file_ref,'_no_noise - test files: ',test_f];
fprintf(v),fprintf('\n')
% compute  averages
RR=mean(rr);UR=mean(ur);BR=mean(br);TR=mean(tr);MR=mean(mr);
eval(['ur_',file_ref,'_',file_test,'=ur;']);
eval(['UR_',file_ref,'_',file_test,'=UR;']);
eval(['min_UR_',file_ref,'_',file_test,'=min(ur);']);
eval(['max_UR_',file_ref,'_',file_test,'=max(ur);']);

eval(['rr_',file_ref,'_',file_test,'=rr;']);
eval(['RR_',file_ref,'_',file_test,'=RR;']);
eval(['min_RR_',file_ref,'_',file_test,'=min(rr);']);
eval(['max_RR_',file_ref,'_',file_test,'=max(rr);']);

eval(['br_',file_ref,'_',file_test,'=br;']);
eval(['BR_',file_ref,'_',file_test,'=BR;']);
eval(['min_BR_',file_ref,'_',file_test,'=min(br);']);
eval(['max_BR_',file_ref,'_',file_test,'=max(br);']);

eval(['tr_',file_ref,'_',file_test,'=tr;']);
eval(['TR_',file_ref,'_',file_test,'=TR;']);
eval(['min_TR_',file_ref,'_',file_test,'=min(tr);']);
eval(['max_TR_',file_ref,'_',file_test,'=max(tr);']);

fprintf('RR: %g  UR: %g  BR: %g \n',RR,UR,BR)
fprintf('TR: %g  MR: %g\n\n',TR,MR)
RRs=std(rr);URs=std(ur);BRs=std(br);TRs=std(tr);MRs=std(mr);

eval(['URs_',file_ref,'_',file_test,'=URs;']);
eval(['RRs_',file_ref,'_',file_test,'=RRs;']);
eval(['BRs_',file_ref,'_',file_test,'=BRs;']);
eval(['TRs_',file_ref,'_',file_test,'=TRs;']);

fprintf('RRs: %g  URs: %g  BRs: %g \n',RRs,URs,BRs)
fprintf('TRs: %g  MRs: %g\n',TRs,MRs)
```

```
if iplot==1,
clg
t=['ref. nb'];
%ref_f=num2str(file_ref);
test_f=num2str(test_fil);
subplot(221),plot(ur)
xlabel('ref. nb')
title('cross-correlation 1 - ur')
subplot(222),plot(rr)
xlabel('ref. nb')
title('cross-correlation 2 - rr')
subplot(223),plot(br)
text(.21,.035,t,'sc')
title('Bhattacharya dist - br')
subplot(224),plot(tr)
text(.71,.035,t,'sc')
title('Divergence - tr')
%text(.892,0,date2,'sc')
if iwin==0,
text(.45,.5,'rect. window','sc')
elseif iwin==1,text(.45,.5,'hamming window','sc');end
v=['ref. file: ',file_ref,'_no_noise - test files: ',test_f];
text(.12,0,v,'sc')
end
end

end
clear ur y rr br tr mr x ref refn refd xeps yeps RUNS Runs file_ref;
clear ref0 file_test x iplot iwin ref_fil test_fil isq ref i ur y;
clear ref_f test_f v b n UR URs TR TRs Run RRs RR MRs MR BR BRs t;
```

C.4: Function Noisy_Sig.M

```
function noisy=noisy_sig(RAW_DATA,db);
 if db==0          % Scale for noise
    scale=1;
  elseif db==1;
     scale=sqrt(1.26);
  elseif db==2;
     scale=sqrt(1.6);
elseif db==3;
     scale=sqrt(2);
elseif db==4;
     scale=sqrt(2.5);
elseif db==5;
     scale=sqrt(3.2);
elseif db==6;
     scale=sqrt(4);
elseif db==7;
     scale=sqrt(5);
elseif db==8;
     scale=sqrt(6.3);
elseif db==9;
     scale=sqrt(8);
```

```
elseif db==10;
     scale=sqrt(10);
elseif db==11;
     scale=sqrt(12.6);
elseif db==12;
     scale=sqrt(16);
elseif db==13;
     scale=sqrt(20);
elseif db==14;
     scale=sqrt(25);
elseif db==15;
     scale=sqrt(32);
elseif db==16;
     scale=sqrt(40);
elseif db==17;
     scale=sqrt(50);
elseif db==18;
     scale=sqrt(63);
elseif db==19;
     scale=sqrt(80);
elseif db==20;
     scale=sqrt(100);
 end
 rand('normal');
len=length(RAW_DATA);
segments=fix(len/1020);
 for i=1:segments+1
          A=(((i-1)*1020)+1);
          B=(i*1020);
         C=len;
  if i<segments+1
pwr(i)=(sum((RAW_DATA((A:B))).^2))/1020;
noisy_sig(A:B)=(RAW_DATA(A:B))+(sqrt(pwr(i))*scale*rand(1,1020)');
 elseif i==segments+1
pwr(i)=(sum((RAW_DATA(A:C)).^2))/1020;
seg=length(RAW_DATA(A:C));
noisy_sig(A:C)=(RAW_DATA(A:C))+(sqrt(pwr(i))*scale*rand(1,seg)');
 end
end
noisy=noisy_sig';
```

C.5: Function Noisy_X1_fcn.M

```
function
ans=noisy_X1_fcn(pow_200_noise,pow_210_noise,pow_220_noise,pow_230_noise)
ans=[pow_200_noise;
pow_210_noise(1,:);
pow_210_noise(3,:);
pow_210_noise(4,:);
pow_210_noise(5,:);
pow_210_noise(6,:);
pow_230_noise(1,:);
pow_230_noise(2,:);
pow_230_noise(3,:);
pow_230_noise(5,:);
pow_230_noise(6,:);
pow_230_noise(7,:);
pow_230_noise(8,:);
pow_230_noise(9,:)];
```

C.6: Function Noisy_X2_fcn.M

```
function ans=noisy_X2_fcn(pow_220_noise)
ans=[pow_220_noise(1,:);
pow_220_noise(2,:);
pow_220_noise(4,:);
pow_220_noise(5,:);
pow_220_noise(6,:)];
```

C.7: Function Noisy_X3_fcn.M

```
function ans=noisy_X3_fcn(pow_210_noise,pow_220_noise,pow_230_noise)
ans=[pow_210_noise(2,:);
pow_220_noise(3,:);
pow_230_noise(4,:);
pow_230_noise(10,:);
pow_230_noise(11,:);
pow_230_noise(12,:);
pow_230_noise(13,:)];
```

B. MULTIPLE RUN PROGRAMS

C.8: Program BIG.M

```
%**************************************************************
%       SYSTEM CHECK PROGRAM
%**************************************************************
clg;
clear;
randn('seed',0);
BIG_adder_clear;
for loopy = 1:100;
['Top of Loopy loop']
loopy
BIG_no_noise_lofar;
['finished ,BIG_no_noise_LOFAR']
BIG_noisy_LOFAR;
['finished ,BIG_noisy_LOFAR']
BIG_DIST_all_var;
BIG_band2_plots;
```

```
['finished BIG_band_plots']
BIG_summer;
['finished BIG_summer']
        mark=num2str(loopy);
       ['did the mark']
       mark
BIG_var_save
       clear
       eval(['load PLACE_MARK']); %this saves the place mark
       ['Just loaded PLACE_MARK after the clear']
       eval(['load GRAPH_',mark]);
end
BIG_average;
['finished BIG_average']
BIG_plots;
```

C.9: Program BIG_adder_clear.M

```
%*******************************************************************
%      This clears the adding matrix
%*******************************************************************
   adder_b=zeros(3,10);
adder_c=zeros(3,10);
adder_d=zeros(3,10);
adder_e=zeros(3,10);
adder_b1=zeros(3,10);
adder_c1=zeros(3,10);
adder_d1=zeros(3,10);
adder_e1=zeros(3,10);
adder_b2=zeros(3,10);
adder_c2=zeros(3,10);
adder_d2=zeros(3,10);
adder_e2=zeros(3,10);

  adder_bb=zeros(2,10);
adder_cc=zeros(2,10);
adder_dd=zeros(2,10);
adder_ee=zeros(2,10);
adder_bb1=zeros(2,10);
adder_cc1=zeros(2,10);
adder_dd1=zeros(2,10);
adder_ee1=zeros(2,10);
adder_bb2=zeros(2,10);
adder_cc2=zeros(2,10);
adder_dd2=zeros(2,10);
adder_ee2=zeros(2,10);

  adder_bbb=zeros(1,10);
adder_ccc=zeros(1,10);
adder_ddd=zeros(1,10);
adder_eee=zeros(1,10);
adder_bbb1=zeros(1,10);
adder_ccc1=zeros(1,10);
adder_ddd1=zeros(1,10);
```

```
adder_eee1=zeros(1,10);
adder_bbb2=zeros(1,10);
adder_ccc2=zeros(1,10);
adder_ddd2=zeros(1,10);
adder_eee2=zeros(1,10);

  adder_bbbb=zeros(2,10);
adder_cccc=zeros(2,10);
adder_dddd=zeros(2,10);
adder_eeee=zeros(2,10);
adder_bbbb1=zeros(2,10);
adder_cccc1=zeros(2,10);
adder_dddd1=zeros(2,10);
adder_eeee1=zeros(2,10);
adder_bbbb2=zeros(2,10);
adder_cccc2=zeros(2,10);
adder_dddd2=zeros(2,10);
adder_eeee2=zeros(2,10);
```

C.10: Program BIG_band2_plots.M

```
%              BIG_band2_plots;

%    ********************************************
% THIS PROGRAM CALCULATES AND PLOTS CURVES
% BASED ON RESULTS FROM THE CORRELATIONS
%    ********************************************

    % THIS PORTION HANDLES THE no_noise SITUATIONS
for x=1:3
   xx=num2str(x);

  b(x,1)=eval(['UR_X',xx,'_X',xx,'_no_noise']);        % this is for avg #
  c(x,1)=eval(['RR_X',xx,'_X',xx,'_no_noise']);
  d(x,1)=eval(['TR_X',xx,'_X',xx,'_no_noise']);
  e(x,1)=eval(['BR_X',xx,'_X',xx,'_no_noise']);

  b1(x,1)=eval(['min_UR_X',xx,'_X',xx,'_no_noise']);  % this is for min #
  c1(x,1)=eval(['min_RR_X',xx,'_X',xx,'_no_noise']);
  d1(x,1)=eval(['min_TR_X',xx,'_X',xx,'_no_noise']);
  e1(x,1)=eval(['min_BR_X',xx,'_X',xx,'_no_noise']);

  b2(x,1)=eval(['max_UR_X',xx,'_X',xx,'_no_noise']);   % this is for max #
  c2(x,1)=eval(['max_RR_X',xx,'_X',xx,'_no_noise']);
  d2(x,1)=eval(['max_TR_X',xx,'_X',xx,'_no_noise']);
  e2(x,1)=eval(['max_BR_X',xx,'_X',xx,'_no_noise']);
end

    % THIS PORTION HANDLES ALL THE REST
for x=1:3;
for y=1:9;
    xx=num2str(x);
    mat=[0 1 2 3 4 5 10 15 20];
    yy=num2str(mat(y));
   b(x,y+1)=eval(['UR_X',xx,'_X',xx,'_',yy,'db']);
   c(x,y+1)=eval(['RR_X',xx,'_X',xx,'_',yy,'db']);
   d(x,y+1)=eval(['TR_X',xx,'_X',xx,'_',yy,'db']);
   e(x,y+1)=eval(['BR_X',xx,'_X',xx,'_',yy,'db']);

   b1(x,y+1)=eval(['min_UR_X',xx,'_X',xx,'_',yy,'db']);
   c1(x,y+1)=eval(['min_RR_X',xx,'_X',xx,'_',yy,'db']);
   d1(x,y+1)=eval(['min_TR_X',xx,'_X',xx,'_',yy,'db']);
   e1(x,y+1)=eval(['min_BR_X',xx,'_X',xx,'_',yy,'db']);

   b2(x,y+1)=eval(['max_UR_X',xx,'_X',xx,'_',yy,'db']);
   c2(x,y+1)=eval(['max_RR_X',xx,'_X',xx,'_',yy,'db']);
   d2(x,y+1)=eval(['max_TR_X',xx,'_X',xx,'_',yy,'db']);
   e2(x,y+1)=eval(['max_BR_X',xx,'_X',xx,'_',yy,'db']);
end
end
%      ***********************************************
% THIS PART SORTS X1 WITH X2 & X2 WITH X3
%      ***********************************************

    % THIS PORTION HANDLES THE no_noise SITUATIONS

for x=1:2;
 xx=num2str(x+1);
 xy=num2str(x);

   bb(x,1)=eval(['UR_X',xy,'_X',xx,'_no_noise';]);
   cc(x,1)=eval(['RR_X',xy,'_X',xx,'_no_noise';]);
   dd(x,1)=eval(['TR_X',xy,'_X',xx,'_no_noise';]);
   ee(x,1)=eval(['BR_X',xy,'_X',xx,'_no_noise';]);

   bb1(x,1)=eval(['min_UR_X',xy,'_X',xx,'_no_noise';]);
   cc1(x,1)=eval(['min_RR_X',xy,'_X',xx,'_no_noise';]);
   dd1(x,1)=eval(['min_TR_X',xy,'_X',xx,'_no_noise';]);
   ee1(x,1)=eval(['min_BR_X',xy,'_X',xx,'_no_noise';]);

   bb2(x,1)=eval(['max_UR_X',xy,'_X',xx,'_no_noise';]);
   cc2(x,1)=eval(['max_RR_X',xy,'_X',xx,'_no_noise';]);
   dd2(x,1)=eval(['max_TR_X',xy,'_X',xx,'_no_noise';]);
   ee2(x,1)=eval(['max_BR_X',xy,'_X',xx,'_no_noise';]);
end


for x=1:2
for y=1:9


     xx=num2str(x);
     xxx=num2str(x+1);
     mat=[0 1 2 3 4 5 10 15 20];
     yy=num2str(mat(y));

   bb(x,y+1)=eval(['UR_X',xx,'_X',xxx,'_',yy,'db';]);
   cc(x,y+1)=eval(['RR_X',xx,'_X',xxx,'_',yy,'db';]);
```

```
  dd(x,y+1)=eval(['TR_X',xx,'_X',xxx,'_',yy,'db';]);
  ee(x,y+1)=eval(['BR_X',xx,'_X',xxx,'_',yy,'db';]);

  bb1(x,y+1)=eval(['min_UR_X',xx,'_X',xxx,'_',yy,'db';]);
  cc1(x,y+1)=eval(['min_RR_X',xx,'_X',xxx,'_',yy,'db';]);
  dd1(x,y+1)=eval(['min_TR_X',xx,'_X',xxx,'_',yy,'db';]);
  ee1(x,y+1)=eval(['min_BR_X',xx,'_X',xxx,'_',yy,'db';]);

  bb2(x,y+1)=eval(['max_UR_X',xx,'_X',xxx,'_',yy,'db';]);
  cc2(x,y+1)=eval(['max_RR_X',xx,'_X',xxx,'_',yy,'db';]);
  dd2(x,y+1)=eval(['max_TR_X',xx,'_X',xxx,'_',yy,'db';]);
  ee2(x,y+1)=eval(['max_BR_X',xx,'_X',xxx,'_',yy,'db';]);
end
end

%   ************************************************
% THIS PART SORTS X1 WITH X3
%   ************************************************

  bbb(1,1)=min_UR_X1_X3_no_noise;
  ccc(1,1)=min_RR_X1_X3_no_noise;
  ddd(1,1)=min_TR_X1_X3_no_noise;
  eee(1,1)=min_BR_X1_X3_no_noise;

  bbb1(1,1)=min_UR_X1_X3_no_noise;
  ccc1(1,1)=min_RR_X1_X3_no_noise;
  ddd1(1,1)=min_TR_X1_X3_no_noise;
  eee1(1,1)=min_BR_X1_X3_no_noise;

  bbb2(1,1)=max_UR_X1_X3_no_noise;
  ccc2(1,1)=max_RR_X1_X3_no_noise;
  ddd2(1,1)=max_TR_X1_X3_no_noise;
  eee2(1,1)=max_BR_X1_X3_no_noise;

for x=1;
for y=2:10;
xx=num2str(x);
xxx=num2str(x+2);
mat=[0 0 1 2 3 4 5 10 15 20];
yy=num2str(mat(y));
  bbb(x,y)=eval(['UR_X',xx,'_X',xxx,'_',yy,'db';]);
  ccc(x,y)=eval(['RR_X',xx,'_X',xxx,'_',yy,'db';]);
  ddd(x,y)=eval(['TR_X',xx,'_X',xxx,'_',yy,'db';]);
  eee(x,y)=eval(['BR_X',xx,'_X',xxx,'_',yy,'db';]);

  bbb1(x,y)=eval(['min_UR_X',xx,'_X',xxx,'_',yy,'db';]);
  ccc1(x,y)=eval(['min_RR_X',xx,'_X',xxx,'_',yy,'db';]);
  ddd1(x,y)=eval(['min_TR_X',xx,'_X',xxx,'_',yy,'db';]);
  eee1(x,y)=eval(['min_BR_X',xx,'_X',xxx,'_',yy,'db';]);

  bbb2(x,y)=eval(['max_UR_X',xx,'_X',xxx,'_',yy,'db';]);
  ccc2(x,y)=eval(['max_RR_X',xx,'_X',xxx,'_',yy,'db';]);
  ddd2(x,y)=eval(['max_TR_X',xx,'_X',xxx,'_',yy,'db';]);
  eee2(x,y)=eval(['max_BR_X',xx,'_X',xxx,'_',yy,'db';]);
end
```

```
end

%     *********************************************
% THIS PART SORTS X2 WITH X1
%     *********************************************

   bbbb(1,1)=min_UR_X2_X1_no_noise;
   cccc(1,1)=min_RR_X2_X1_no_noise;
   dddd(1,1)=min_TR_X2_X1_no_noise;
   eeee(1,1)=min_BR_X2_X1_no_noise;

   bbbb1(1,1)=min_UR_X2_X1_no_noise;
   cccc1(1,1)=min_RR_X2_X1_no_noise;
   dddd1(1,1)=min_TR_X2_X1_no_noise;
   eeee1(1,1)=min_BR_X2_X1_no_noise;

   bbbb2(1,1)=max_UR_X2_X1_no_noise;
   cccc2(1,1)=max_RR_X2_X1_no_noise;
   dddd2(1,1)=max_TR_X2_X1_no_noise;
   eeee2(1,1)=max_BR_X2_X1_no_noise;

for x=2;
for y=2:10;
xx=num2str(x);
xxx=num2str(x-1);
mat=[0 0 1 2 3 4 5 10 15 20];
yy=num2str(mat(y));
  bbbb(1,y)=eval(['UR_X',xx,'_X',xxx,'_',yy,'db';]);
  cccc(1,y)=eval(['RR_X',xx,'_X',xxx,'_',yy,'db';]);
  dddd(1,y)=eval(['TR_X',xx,'_X',xxx,'_',yy,'db';]);
  eeee(1,y)=eval(['BR_X',xx,'_X',xxx,'_',yy,'db';]);

  bbbb1(1,y)=eval(['min_UR_X',xx,'_X',xxx,'_',yy,'db';]);
  cccc1(x,y)=eval(['min_RR_X',xx,'_X',xxx,'_',yy,'db';]);
  dddd1(1,y)=eval(['min_TR_X',xx,'_X',xxx,'_',yy,'db';]);
  eeee1(1,y)=eval(['min_BR_X',xx,'_X',xxx,'_',yy,'db';]);
  bbbb2(1,y)=eval(['max_UR_X',xx,'_X',xxx,'_',yy,'db';]);
  cccc2(1,y)=eval(['max_RR_X',xx,'_X',xxx,'_',yy,'db';]);
  dddd2(1,y)=eval(['max_TR_X',xx,'_X',xxx,'_',yy,'db';]);
  eeee2(1,y)=eval(['max_BR_X',xx,'_X',xxx,'_',yy,'db';]);
end
end
```

C.11: Program BIG_summer.M

```
%      BIG_summer;

%*******************************************************
%      This program finds the sum of all plot variables
%*******************************************************

for loop1=1:3;
for loop2=1:10;
```

```
adder_b(loop1,loop2)=b(loop1,loop2)+adder_b(loop1,loop2);
adder_c(loop1,loop2)=c(loop1,loop2)+adder_c(loop1,loop2);
adder_d(loop1,loop2)=d(loop1,loop2)+adder_d(loop1,loop2);
adder_e(loop1,loop2)=e(loop1,loop2)+adder_e(loop1,loop2);
adder_b1(loop1,loop2)=b1(loop1,loop2)+adder_b1(loop1,loop2);
adder_c1(loop1,loop2)=c1(loop1,loop2)+adder_c1(loop1,loop2);
adder_d1(loop1,loop2)=d1(loop1,loop2)+adder_d1(loop1,loop2);
adder_e1(loop1,loop2)=e1(loop1,loop2)+adder_e1(loop1,loop2);
adder_b2(loop1,loop2)=b2(loop1,loop2)+adder_b2(loop1,loop2);
adder_c2(loop1,loop2)=c2(loop1,loop2)+adder_c2(loop1,loop2);
adder_d2(loop1,loop2)=d2(loop1,loop2)+adder_d2(loop1,loop2);
adder_e2(loop1,loop2)=e2(loop1,loop2)+adder_e2(loop1,loop2);
end
end

for loop3=1:2;
for loop4=1:10;
   adder_bb(loop3,loop4)=bb(loop3,loop4)+adder_bb(loop3,loop4);
adder_cc(loop3,loop4)=cc(loop3,loop4)+adder_cc(loop3,loop4);
adder_dd(loop3,loop4)=dd(loop3,loop4)+adder_dd(loop3,loop4);
adder_ee(loop3,loop4)=ee(loop3,loop4)+adder_ee(loop3,loop4);
adder_bb1(loop3,loop4)=bb1(loop3,loop4)+adder_bb1(loop3,loop4);
adder_cc1(loop3,loop4)=cc1(loop3,loop4)+adder_cc1(loop3,loop4);
adder_dd1(loop3,loop4)=dd1(loop3,loop4)+adder_dd1(loop3,loop4);
adder_ee1(loop3,loop4)=ee1(loop3,loop4)+adder_ee1(loop3,loop4);
adder_bb2(loop3,loop4)=bb2(loop3,loop4)+adder_bb2(loop3,loop4);
adder_cc2(loop3,loop4)=cc2(loop3,loop4)+adder_cc2(loop3,loop4);
adder_dd2(loop3,loop4)=dd2(loop3,loop4)+adder_dd2(loop3,loop4);
adder_ee2(loop3,loop4)=ee2(loop3,loop4)+adder_ee2(loop3,loop4);
end
end

for loop5=1:1;
for loop6=1:10;
   adder_bbb(loop5,loop6)=bbb(loop5,loop6)+adder_bbb(loop5,loop6);
adder_ccc(loop5,loop6)=ccc(loop5,loop6)+adder_ccc(loop5,loop6);
adder_ddd(loop5,loop6)=ddd(loop5,loop6)+adder_ddd(loop5,loop6);
adder_eee(loop5,loop6)=eee(loop5,loop6)+adder_eee(loop5,loop6);
adder_bbb1(loop5,loop6)=bbb1(loop5,loop6)+adder_bbb1(loop5,loop6);
adder_ccc1(loop5,loop6)=ccc1(loop5,loop6)+adder_ccc1(loop5,loop6);
adder_ddd1(loop5,loop6)=ddd1(loop5,loop6)+adder_ddd1(loop5,loop6);
adder_eee1(loop5,loop6)=eee1(loop5,loop6)+adder_eee1(loop5,loop6);
adder_bbb2(loop5,loop6)=bbb2(loop5,loop6)+adder_bbb2(loop5,loop6);
adder_ccc2(loop5,loop6)=ccc2(loop5,loop6)+adder_ccc2(loop5,loop6);
adder_ddd2(loop5,loop6)=ddd2(loop5,loop6)+adder_ddd2(loop5,loop6);
adder_eee2(loop5,loop6)=eee2(loop5,loop6)+adder_eee2(loop5,loop6);
end
end

for loop7=1:1;
for loop8=1:10;
   adder_bbbb(loop7,loop8)=bbbb(loop7,loop8)+adder_bbbb(loop7,loop8);
adder_cccc(loop7,loop8)=cccc(loop7,loop8)+adder_cccc(loop7,loop8);
adder_dddd(loop7,loop8)=dddd(loop7,loop8)+adder_dddd(loop7,loop8);
adder_eeee(loop7,loop8)=eeee(loop7,loop8)+adder_eeee(loop7,loop8);
```

```
adder_bbbb1(loop7,loop8)=bbbb1(loop7,loop8)+adder_bbbb1(loop7,loop8);
adder_cccc1(loop7,loop8)=cccc1(loop7,loop8)+adder_cccc1(loop7,loop8);
adder_dddd1(loop7,loop8)=dddd1(loop7,loop8)+adder_dddd1(loop7,loop8);
adder_eeee1(loop7,loop8)=eeee1(loop7,loop8)+adder_eeee1(loop7,loop8);
adder_bbbb2(loop7,loop8)=bbbb2(loop7,loop8)+adder_bbbb2(loop7,loop8);
adder_cccc2(loop7,loop8)=cccc2(loop7,loop8)+adder_cccc2(loop7,loop8);
adder_dddd2(loop7,loop8)=dddd2(loop7,loop8)+adder_dddd2(loop7,loop8);
adder_eeee2(loop7,loop8)=eeee2(loop7,loop8)+adder_eeee2(loop7,loop8);
end
end
clear b c d e b1 c1 d1 e1 b2 c2 d2 e2 bb cc dd ee bb1 cc1 dd1 ee1
clear bb2 cc2 dd2 ee2 bbb ccc ddd eee bbb1 ccc1 ddd1 eee1
clear bbb2 ccc2 eee2 ddd2 bbbb cccc dddd eeee bbbb1 cccc1 dddd1 eeee1
clear bbbb2 cccc2 dddd2 eeee2
```

C.12: Program BIG_var_save.M

```
% THIS PROGRAM WILL SAVE ALL IMPORTANT VARIABLES %

eval(['save GRAPH_',mark,' loopy adder_b adder_c adder_d adder_e adder_b1 adder_c1
adder_d1 adder_e1 adder_b2 adder_c2 adder_d2 adder_e2 adder_bb adder_cc adder_dd
adder_ee adder_bb1 adder_cc1 adder_dd1 adder_ee1 adder_bb2 adder_cc2 adder_dd2
adder_ee2 adder_bbb adder_ccc adder_ddd adder_eee adder_bbb1 adder_ccc1 adder_ddd1
adder_eee1 adder_bbb2 adder_ccc2 adder_ddd2 adder_eee2 adder_bbbb adder_cccc
adder_dddd adder_eeee adder_bbbb1 adder_cccc1 adder_dddd1 adder_eeee1 adder_bbbb2
adder_cccc2 adder_dddd2 adder_eeee2']);

if loopy==1
eval(['save RUN_',mark,' X1_0db X1_1db X1_2db X1_3db X1_4db X1_5db X1_10db
%X1_15db X1_20db X2_0db X2_1db X2_2db X2_3db X2_4db X2_5db X2_10db
X2_15db %X2_20db X3_0db X3_1db X3_2db X3_3db X3_4db X3_5db X3_10db
X3_15db X3_20db']);

elseif loopy==20
eval(['save RUN_',mark,' X1_0db X1_1db X1_2db X1_3db X1_4db X1_5db X1_10db
%X1_15db X1_20db X2_0db X2_1db X2_2db X2_3db X2_4db X2_5db X2_10db
X2_15db %X2_20db X3_0db X3_1db X3_2db X3_3db X3_4db X3_5db X3_10db
X3_15db X3_20db']);

elseif loopy==40
eval(['save RUN_',mark,' X1_0db X1_1db X1_2db X1_3db X1_4db X1_5db X1_10db
%X1_15db X1_20db X2_0db X2_1db X2_2db X2_3db X2_4db X2_5db X2_10db
X2_15db %X2_20db X3_0db X3_1db X3_2db X3_3db X3_4db X3_5db X3_10db
X3_15db X3_20db']);

elseif loopy==40
eval(['save RUN_',mark,' X1_0db X1_1db X1_2db X1_3db X1_4db X1_5db X1_10db
%X1_15db X1_20db X2_0db X2_1db X2_2db X2_3db X2_4db X2_5db X2_10db
X2_15db %X2_20db X3_0db X3_1db X3_2db X3_3db X3_4db X3_5db X3_10db
X3_15db X3_20db']);

elseif loopy==60
```

```
eval(['save RUN_',mark,' X1_0db X1_1db X1_2db X1_3db X1_4db X1_5db X1_10db
%X1_15db X1_20db X2_0db X2_1db X2_2db X2_3db X2_4db X2_5db X2_10db
X2_15db %X2_20db X3_0db X3_1db X3_2db X3_3db X3_4db X3_5db X3_10db
X3_15db X3_20db']);

elseif loopy==80
eval(['save RUN_',mark,' X1_0db X1_1db X1_2db X1_3db X1_4db X1_5db X1_10db
%X1_15db X1_20db X2_0db X2_1db X2_2db X2_3db X2_4db X2_5db X2_10db
X2_15db %X2_20db X3_0db X3_1db X3_2db X3_3db X3_4db X3_5db X3_10db
X3_15db X3_20db']);

elseif loopy==100
eval(['save RUN_',mark,' X1_0db X1_1db X1_2db X1_3db X1_4db X1_5db X1_10db
%X1_15db X1_20db X2_0db X2_1db X2_2db X2_3db X2_4db X2_5db X2_10db
X2_15db %X2_20db X3_0db X3_1db X3_2db X3_3db X3_4db X3_5db X3_10db
X3_15db X3_20db']);

end
eval(['save PLACE_MARK mark']);
```

C.13: Program BIG_average.M

```
%   BIG_average

%**********************************************************
%       This program takes the average for the plots
%**********************************************************

b=adder_b/loopy;
c=adder_c/loopy;
d=adder_d/loopy;
e=adder_e/loopy;
b1=adder_b1/loopy;
c1=adder_c1/loopy;
d1=adder_d1/loopy;
e1=adder_e1/loopy;
b2=adder_b2/loopy;
c2=adder_c2/loopy;
d2=adder_d2/loopy;
e2=adder_e2/loopy;

bb=adder_bb/loopy;
cc=adder_cc/loopy;
dd=adder_dd/loopy;
ee=adder_ee/loopy;
bb1=adder_bb1/loopy;
cc1=adder_cc1/loopy;
dd1=adder_dd1/loopy;
ee1=adder_ee1/loopy;
bb2=adder_bb2/loopy;
cc2=adder_cc2/loopy;
dd2=adder_dd2/loopy;
ee2=adder_ee2/loopy;
```

```
bbb=adder_bbb/loopy;
ccc=adder_ccc/loopy;
ddd=adder_ddd/loopy;
eee=adder_eee/loopy;
bbb1=adder_bbb1/loopy;
ccc1=adder_ccc1/loopy;
ddd1=adder_ddd1/loopy;
eee1=adder_eee1/loopy;
bbb2=adder_bbb2/loopy;
ccc2=adder_ccc2/loopy;
ddd2=adder_ddd2/loopy;
eee2=adder_eee2/loopy;

bbbb=adder_bbbb/loopy;
cccc=adder_cccc/loopy;
dddd=adder_dddd/loopy;
eeee=adder_eeee/loopy;
bbbb1=adder_bbbb1/loopy;
cccc1=adder_cccc1/loopy;
dddd1=adder_dddd1/loopy;
eeee1=adder_eeee1/loopy;
bbbb2=adder_bbbb2/loopy;
cccc2=adder_cccc2/loopy;
dddd2=adder_dddd2/loopy;
eeee2=adder_eeee2/loopy;
```

C.14: Program BIG_plots.M

```
%       BIG_plots;

%  **********************************************
%  THIS PROGRAM CALCULATES AND PLOTS CURVES
%  BASED ON RESULTS FROM THE CORRELATIONS
%  **********************************************

range=[0 -1 -2 -3 -4 -5 -10 -15 -20];
db_len=2:10;
xrange=-20:0;
yrange=ones(1,21);


band=ones(1,5);
range1=[0 -1 -2 -3 -4 -5 -10 -15 -20];
range2=[0 -1 -2 -3 -4 -5 -10 -15 -20];
range3=[0 -1 -2 -3 -4 -5 -10 -15 -20];

%range2=[-.33 -1.33 -2.33 -3.33 -4.33 -5.33 -10.33 -15.33 -19.33];
%range3=[-.66 -1.66 -2.66 -3.66 -4.66 -5.66 -10.66 -15.66 -19.66];

% THIS PLOTS X1,X2,X3 with themselves for none-20db
% for normalized cross coorelation (1st plot)
%UR
clg
hold off;
```

```
%   ************************************************************
% PLOT       SET #1    "X1 w/ X1" & "X2 w/ X2" & "X3 w/ X3"
%   *************************************************************

%%%%%%Cross Correlation Plots w/ DC Effects %%%%%
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(221)

axis ([-20 0 0 1])
plot(range,b(1,db_len),'-');
hold on;
plot(xrange,(b(1,1)*yrange),'-');

plot(range,b(2,db_len),'--');
plot(xrange,(b(2,1)*yrange),'--');

plot(range,b(3,db_len),':');
plot(xrange,(b(3,1)*yrange),':');
text(-11,.4,'__=X1_X1#db')
text(-11,.3,'---=X2_X2#db')
text(-11,.2,'...=X3_X3#db')
%text(-8,.1,'Straight line = no_noise level')
title('Cross Corr.')
xlabel('dB')
ylabel('Corr. Coef.')
hold off

%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)
axis ([-20 0 0 1])
plot(range,b(1,db_len),'-');
hold on;
plot(xrange,(b(1,1)*yrange),'-');
for loop=1:9;
    plot(band*range1(loop),linspace(b1(1,loop+1),b2(1,loop+1),5));
end
text(-9,.3,'X1_X1#db')
%text(-8,.1,'Straight line = no_noise level')
title('X1 Band')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%PLOT#3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)
axis ([-20 0 0 1])
plot(range,b(2,db_len),'--');
hold on
plot(xrange,(b(2,1)*yrange),'--');
for loop=1:9;
    plot(band*range2(loop),linspace(b1(2,loop+1),b2(2,loop+1),5),'--');
end
text(-19,.8,'X2_X2#db')
%text(-8,.1,'Straight line = no_noise level')
title('X2 Band')
```

```
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)
axis ([-20 0 0 1])
plot(range,b(3,db_len),':');
hold on
plot(xrange,(b(3,1)*yrange),':');
for loop=1:9;
    plot(band*range3(loop),linspace(b1(3,loop+1),b2(3,loop+1),5),':');
end
text(-19,.8,'X3_X3#db')
%text(-8,.1,'Straight line = no_noise level')
title('X3 Band')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%text(0.3,0,' EACH TEMPLATE COMPARED WITH ITSELF WITH NOISE ','sc')
pause
%%%%%%% Cross Correlation Plots w/no DC effects %%%%%%%
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% THIS PLOTS X1,X2,X3 with themselves for none-20db
% for NORMALIZED CROSS CORRELATION COEFFICIENT (2nd plot)
%RR
clg
subplot(221)
axis ([-20 0 0 1])

plot(range,c(1,db_len),'-');
hold on
plot(xrange,(c(1,1)*yrange),'-');

plot(range,c(2,db_len),'--');
plot(xrange,(c(2,1)*yrange),'--');

plot(range,c(3,db_len),':');
plot(xrange,(c(3,1)*yrange),':');

text(-11,.4,'__=X1_X1#db')
text(-11,.3,'---=X2_X2#db')
text(-11,.2,'...=X3_X3#db')
%text(-8,.1,'Straight line = no_noise level')
title('Cross Corr.(no mean)')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%text(0.3,0,' EACH TEMPLATE COMPARED WITH ITSELF WITH NOISE ','sc')

%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)

axis ([-20 0 0 1])
plot(range,c(1,db_len),'-');
```

```
hold on;
plot(xrange,(c(1,1)*yrange),'-');
for loop=1:9;
    plot(band*range1(loop),linspace(c1(1,loop+1),c2(1,loop+1),5));
end
text(-9,.3,'X1_X1#db')
%text(-8,.1,'Straight line = no_noise level')
title('X1 Band')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%PLOT#3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)

axis ([-20 0 0 1])
plot(range,c(2,db_len),'--');
hold on;
plot(xrange,(c(2,1)*yrange),'--');
for loop=1:9;
    plot(band*range1(loop),linspace(c1(2,loop+1),c2(2,loop+1),5),'--');
end
text(-19,.8,'X2_X2#db')
%text(-8,.1,'Straight line = no_noise level')
title('X2 Band')
xlabel('dB')
ylabel('Corr. Coef.')
hold off

%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)

axis ([-20 0 0 1])
plot(range,c(3,db_len),':');
hold on;
plot(xrange,(c(3,1)*yrange),':');
for loop=1:9;
    plot(band*range1(loop),linspace(c1(3,loop+1),c2(3,loop+1),5),':');
end
text(-19,.8,'X3_X3#db')

%text(-8,.1,'Straight line = no_noise level')
title('X3 Band')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
pause

%%%%%%% Bhattacharyya distance %%%%%%%
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS PLOTS X1,X2,X3 with themselves for none-20db
% for the Bhattacharyya distance (4th plot)
%BR
clg
subplot(221)
```

```
axis ([-20 0 0 1])
subplot(221)
plot(range,e(1,db_len),'-');
hold on
plot(xrange,(e(1,1)*yrange),'-');

plot(range,e(2,db_len),'--');
plot(xrange,(e(2,1)*yrange),'--');

plot(range,e(3,db_len),':');
plot(xrange,(e(3,1)*yrange),':');

text(-19,.8,'_=X1_X1#db')
text(-19,.6,'--=X2_X2#db')
text(-19,.15,'..=X3_X3#db')
%text(-8,.2,'Straight line = no_noise level')
title('Bhattacharaya dist')
xlabel('dB')
ylabel('Distance')
hold off;
%text(0.3,0,' EACH TEMPLATE COMPARED WITH ITSELF WITH NOISE ','sc')
%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)

axis ([-20 0 0 1])
plot(range,e(1,db_len),'-');
hold on;
plot(xrange,(e(1,1)*yrange),'-');
for loop=1:9;
   plot(band*range1(loop),linspace(e1(1,loop+1),e2(1,loop+1),5));
end
text(-19,.8,'X1_X1#db')
%text(-8,.1,'Straight line = no_noise level')
title('X1 Band')
xlabel('dB')
ylabel('Distance')
hold off
%PLOT#3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)

axis ([-20 0 0 1])
plot(range,e(2,db_len),'--');
hold on;
plot(xrange,(e(2,1)*yrange),'--');
for loop=1:9;
   plot(band*range1(loop),linspace(e1(2,loop+1),e2(2,loop+1),5),'--');
end
text(-19,.8,'X2_X2#db')
%text(-8,.1,'Straight line = no_noise level')
title('X2 Band')
xlabel('dB')
ylabel('Distance')
hold off
%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)
```

```
axis ([-20 0 0 1])
plot(range,e(3,db_len),':');
hold on;
plot(xrange,(e(3,1)*yrange),':');
for loop=1:9;
    plot(band*range1(loop),linspace(e1(3,loop+1),e2(3,loop+1),5),':');
end
text(-19,.8,'X3_X3#db')
%text(-8,.1,'Straight line = no_noise level')
title('X3 Band')
xlabel('dB')
ylabel('Distance')
hold off
pause
%%%%%%%%%%%%% Divergence %%%%%%%%%%%%%%%
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% THIS PLOTS X1,X2,X3 with themselves for none-20db
% for the divergence (3rd plot)
%TR
clg

subplot(221)
plot(range,d(1,db_len),'-');
axis ([-20 0 0 20])

hold on
plot(xrange,(d(1,1)*yrange),'-');

plot(range,d(2,db_len),'--');
plot(xrange,(d(2,1)*yrange),'--');

plot(range,d(3,db_len),':');
plot(xrange,(d(3,1)*yrange),':');

text(-19,16.5,'__=X1_X1#db')
text(-19,10,'---=X2_X2#db')
text(-19,2,'...=X3_X3#db')
%text(-12,12,'Straight line = no_noise level')
title('Divergence')
xlabel('dB')
ylabel('Divergence')
hold off;
%text(0.3,0,' EACH TEMPLATE COMPARED WITH ITSELF WITH NOISE ','sc')

%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)
plot(range,d(1,db_len),'-');
axis ([-20 0 0 20])
hold on;
plot(xrange,(d(1,1)*yrange),'-');
for loop=1:9;
    plot(band*range1(loop),linspace(d1(1,loop+1),d2(1,loop+1),5));
end
text(-19,17,'X1_X1#db')
```

```
%text(-8,.1,'Straight line = no_noise level')
title('X1 Band')
xlabel('dB')
ylabel('Divergence')
hold off
%PLOT#3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)
plot(range,d(2,db_len),'--');

axis ([-20 0 0 20])
hold on;
plot(xrange,(d(2,1)*yrange),'--');
for loop=1:9;
    plot(band*range1(loop),linspace(d1(2,loop+1),d2(2,loop+1),5),'--');
end
text(-19,17,'X2_X2#db')
%text(-8,.1,'Straight line = no_noise level')
title('X2 Band')
xlabel('dB')
ylabel('Divergence')
hold off
%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)
plot(range,d(3,db_len),':');
axis ([-20 0 0 20])
hold on;
plot(xrange,(d(3,1)*yrange),':');
for loop=1:9;
    plot(band*range1(loop),linspace(d1(3,loop+1),d2(3,loop+1),5),':');
end
text(-19,17,'X3_X3#db')
%text(-8,.1,'Straight line = no_noise level')
title('X3 Band')
xlabel('dB')
ylabel('Divergence')
hold off

pause
%     **********************************************************
% PLOT       SET #2  "X1 w/ X1" & "X1 w/ X2" & "X1 w/ X3"
%     **********************************************************
%%%%%%Cross Correlation w/DC %%%%%%%%
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clg
cla
subplot(221)


%UR
plot(range,b(1,db_len),'-');
axis ([-20 0 0 1])
hold on;
plot(xrange,(b(1,1)*yrange),'-');

plot(range,bb(1,db_len),'--');
plot(xrange,(bb(1,1)*yrange),'--');

plot(range,bbb(1,db_len),':');
plot(xrange,(bbb(1,1)*yrange),':');

text(-19,.8,'__=X1_X1#db')
text(-19,.7,'--=X1_X2#db')
text(-19,.6,': =X1_X3#db')
%text(-19,.7,'Straight line = no_noise level')
title('Cross Corr.')
xlabel('dB')
ylabel('Corr. Coef')
hold off;
%text(0.35,0,' X1 TEMPLATE COMPARED WITH X2 AND X3 ','sc')

%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)

plot(range,b(1,db_len),'-');
axis ([-20 0 0 1])

hold on;
plot(xrange,(b(1,1)*yrange),'-');
for loop=1:9;
    plot(band*range1(loop),linspace(b1(1,loop+1),b2(1,loop+1),5));
end
text(-9,.3,'X1_X1#db')
%text(-8,.1,'Straight line = no_noise level')
title('X1 Band')
xlabel('dB')
ylabel('Corr. Coef.')
hold off

%PLOT#3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)
cla
plot(range,bb(1,db_len),'--');
axis ([-20 0 0 1])

hold on
plot(xrange,(bb(1,1)*yrange),'--');
for loop=1:9;
    plot(band*range1(loop),linspace(bb1(1,loop+1),bb2(1,loop+1),5),'--');
end
text(-19,.7,'X1_X2#db')
title('X2 Band')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)
cla
axis ([-20 0 0 1])
plot(range,bbb(1,db_len),':');
axis ([-20 0 0 1])
```

```
hold on
plot(xrange,(bbb(1,1)*yrange),':');
for loop=1:9;
    plot(band*range1(loop),linspace(bbb1(1,loop+1),bbb2(1,loop+1),5),':');
end
text(-19,.7,'X1_X3#db')
title('X3 Band')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
pause
% THIS PLOTS X1 w/ X2  &  X1 w/ X3 for none-20db
% for the cross correlation (2nd plot)
%RR
%%%%% Cross Correlation w/o DC %%%%%%
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clg
subplot(221)

plot(range,c(1,db_len),'-');
axis ([-20 0 0 1])
hold on;
plot(xrange,(c(1,1)*yrange),'-');

plot(range,ccc(1,db_len),':');
plot(xrange,(ccc(1,1)*yrange),':');

plot(range,cc(1,db_len),'--');
plot(xrange,(cc(1,1)*yrange),'--');

text(-19,.8,'__=X1_X1#db')
text(-19,.7,'--=X1_X2#db')
text(-19,.6,': =X1_X3#db')
%text(-19,.7,'Straight line = no_noise level')
title('Cross Corr. (no mean)')
xlabel('dB')
ylabel('Mean Value')
hold off;

%text(0.35,0,' X1 TEMPLATE COMPARED WITH X2 AND X3 ','sc')
%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)

plot(range,c(1,db_len),'-');
axis ([-20 0 0 1])

hold on;
plot(xrange,(c(1,1)*yrange),'-');
for loop=1:9;
    plot(band*range1(loop),linspace(c1(1,loop+1),c2(1,loop+1),5));
end
text(-9,.3,'X1_X1#db')
%text(-8,.1,'Straight line = no_noise level')
title('X1 Band')
```

```
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%PLOT#3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)
cla
plot(range,cc(1,db_len),'--');
axis ([-20 0 0 1])

hold on
plot(xrange,(cc(1,1)*yrange),'--');
for loop=1:9;
    plot(band*range1(loop),linspace(cc1(1,loop+1),cc2(1,loop+1),5),'--');
end
text(-19,.7,'X1_X2#db')
title('X2 Band')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)
cla
plot(range,ccc(1,db_len),':');
axis ([-20 0 0 1])

hold on
plot(xrange,(ccc(1,1)*yrange),':');
for loop=1:9;
    plot(band*range1(loop),linspace(ccc1(1,loop+1),ccc2(1,loop+1),5),':');
end
text(-19,.7,'X1_X3#db')

title('X3 Band')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
pause
% THIS PLOTS X1 w/ X2  &  X1 w/ X3 for none-20db
% for the Bhattacharyya distance (4th plot)
%BR
%%%%%%%Bhattacharyya distance %%%%%%%%%%%
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clg
subplot(221)
plot(range,e(1,db_len),'-');
axis ([-20 0 0 1])

hold on;
plot(xrange,(e(1,1)*yrange),'-');

plot(range,eee(1,db_len),':');
plot(xrange,(eee(1,1)*yrange),':');

plot(range,ee(1,db_len),'--');
plot(xrange,(ee(1,1)*yrange),'--');
```

```
text(-19,.9,'__=X1_X1#db')
text(-19,.8,'--=X1_X2#db')
text(-19,.7,': =X1_ X3#db')
%text(-19,.45,'Straight line = no_noise level')
title('Bhattacharyya dist ')
xlabel('dB')
ylabel('Distance')
hold off;
%text(0.35,0,' X1 TEMPLATE COMPARED WITH X2 AND X3 ','sc')
%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)

plot(range,e(1,db_len),'-');
axis ([-20 0 0 1])

hold on;
plot(xrange,(e(1,1)*yrange),'-');
for loop=1:9;
    plot(band*range1(loop),linspace(e1(1,loop+1),e2(1,loop+1),5));
end
text(-17,.8,'X1_X1#db')
%text(-8,.1,'Straight line = no_noise level')
title('X1 Band')
xlabel('dB')
ylabel('Distance')
hold off
%PLOT#3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)
cla

plot(range,ee(1,db_len),'--');
axis ([-20 0 0 1])
hold on
plot(xrange,(ee(1,1)*yrange),'--');
for loop=1:9;
    plot(band*range1(loop),linspace(ee1(1,loop+1),ee2(1,loop+1),5),'--');
end
text(-17,.7,'X1_X2#db')
title('X2 Band')
xlabel('dB')
ylabel('Distance')
hold off
%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)
cla
plot(range,ee(2,db_len),':');
axis ([-20 0 0 1])

hold on
plot(xrange,(eee(1,1)*yrange),':');
for loop=1:9;
    plot(band*range1(loop),linspace(eee1(1,loop+1),eee2(1,loop+1),5),':');
end
text(-17,.7,'X1_X3#db')
```

```
title('X3 Band')
xlabel('dB')
ylabel('Distance')
hold off
pause
% THIS PLOTS X1 w/ X2  &  X1 w/ X3 for none-20db
% for the divergence (3rd plot)
%TR
%%%%% Divergence %%%%%%%%%%%%
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clg
subplot(221)
cla
plot(range,d(1,db_len),'-');
axis ([-20 0 0 20])

hold on;
plot(xrange,(d(1,1)*yrange),'-');

plot(range,ddd(1,db_len),':');
plot(xrange,(ddd(1,1)*yrange),':');

plot(range,dd(1,db_len),'--');
plot(xrange,(dd(1,1)*yrange),'--');

text(-19,17,'__=X1_X1#db')
text(-19,14,'--=X1_X2#db')
text(-19,11,': =X1_X3#db')
%text(-19,6,'Straight line = no_noise level')
title('Divergence')
xlabel('dB')
ylabel('Divergence')
hold off;
%text(0.35,0,' X1 TEMPLATE COMPARED WITH X2 AND X3 ','sc')
%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)
cla

plot(range,d(1,db_len),'-');
axis ([-20 0 0 20])
hold on;
plot(xrange,(d(1,1)*yrange),'-');
for loop=1:9;
    plot(band*range1(loop),linspace(d1(1,loop+1),d2(1,loop+1),5));
end
text(-19,17,'X1_X1#db')
%text(-8,.1,'Straight line = no_noise level')
title('X1 Band')
xlabel('dB')
ylabel('Divergence')
hold off
%PLOT#3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)
cla
plot(range,dd(1,db_len),'--');
```

```
axis ([-20 0 0 20])

hold on
plot(xrange,(dd(1,1)*yrange),'--');
for loop=1:9;
    plot(band*range1(loop),linspace(dd1(1,loop+1),dd2(1,loop+1),5),'--');
end
text(-19,14,'X1_X2#db')
title('X2 Band')
xlabel('dB')
ylabel('Divergence')
hold off
%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)
cla
plot(range,ddd(1,db_len),':');
axis ([-20 0 0 20])

hold on
plot(xrange,(ddd(1,1)*yrange),':');
for loop=1:9;
    plot(band*range1(loop),linspace(ddd1(1,loop+1),ddd2(1,loop+1),5),':');
end
text(-19,11,'X1_X3#db')
title('X3 Band')
xlabel('dB')
ylabel('Divergence')
hold off
pause
clg
%*********************************************************
% PLOT        SET #3  "X2 w/ X1" & "X2 w/ X2" & "X2 w/ X3"
%    **********************************************************
%%%%%Cross Correlation w/DC %%%%%%%%%%
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%UR
subplot(221)
plot(range,bbb(1,db_len),':');
axis([-20 0 0 1])

hold on
plot(xrange,(bbb(1,1)*yrange),':');

plot(range,bbbb(1,db_len),'-');
plot(xrange,(bbbb(1,1)*yrange),'-');

plot(range,b(2,db_len),'--');
plot(xrange,(b(2,1)*yrange),'--');

text(-10,.2,': =X2_ X3#db')
text(-19.5,.1,'_ =X2_X1#db')
text(-10,.1,'--=X2_X2#db')
%text(-14,.8,'Straight line = no_noise level')
title('Cross Corr. w/DC')
```

```
xlabel('dB')
ylabel('Corr. Coef.')
hold off

%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)
cla
plot(range,bbbb(1,db_len),'-');
axis([-20 0 0 1])

hold on
plot(xrange,(bbbb(1,1)*yrange),'-');
for loop=1:9;
    plot(band*range1(loop),linspace(bbbb1(1,loop+1),bbbb2(1,loop+1),5),'-');
end
title('X1_Band')
text(-19,.8,': =X2_X1#db')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%PLOT#3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)
cla
plot(range,b(2,db_len),'--');
axis([-20 0 0 1])

hold on;
plot(xrange,(b(2,1)*yrange),'--');
for loop=1:9;
    plot(band*range1(loop),linspace(b1(2,loop+1),b2(2,loop+1),5),'--');
end
title('X2_Band')
text(-19,.8,': =X2_X2#db')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)
cla

plot(range,bbb(1,db_len),':');
axis([-20 0 0 1])
hold on
for loop=1:9;
    plot(band*range1(loop),linspace(bbb1(1,loop+1),bbb2(1,loop+1),5),':');
end
plot(xrange,(bbb(1,1)*yrange),':');
title('X3_Band')
text(-19,.8,': =X2_X3#db')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
pause

clg
```

```
%Cross Correlation w/no DC %%%%
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%UR
subplot(221)
cla
plot(range,ccc(1,db_len),':');
axis([-20 0 0 1])

hold on
plot(xrange,(ccc(1,1)*yrange),':');

plot(range,cccc(1,db_len),'-');
plot(xrange,(cccc(1,1)*yrange),'-');

plot(range,c(2,db_len),'--');
plot(xrange,(c(2,1)*yrange),'--');

text(-10,.2,': =X2_ X3#db')
text(-19.5,.1,'_ =X2_X1#db')
text(-10,.1,'--=X2_X2#db')
%text(-17,.9,'Straight line = no_noise level')
title('Cross Corr.(no mean)')
xlabel('dB')
ylabel('Corr. Coef.')
hold off

%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)
cla
plot(range,cccc(1,db_len),'-');
axis([-20 0 0 1])

hold on
plot(xrange,(ccc(1,1)*yrange),':');
for loop=1:9;
    plot(band*range1(loop),linspace(cccc1(1,loop+1),cccc2(1,loop+1),5),'-');
end
title('X1_Band')
text(-19,.8,': =X2_X1#db')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%PLOT#3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)
cla
plot(range,c(2,db_len),'--');
axis([-20 0 0 1])

hold on;
plot(xrange,(c(2,1)*yrange),'--');
for loop=1:9;
    plot(band*range1(loop),linspace(c1(2,loop+1),c2(2,loop+1),5),'--');
end
```

```
title('X2_Band')
text(-19,.8,': =X2_X2#db')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)
cla

plot(range,ccc(1,db_len),':');
axis([-20 0 0 1])
hold on
for loop=1:9;
    plot(band*range1(loop),linspace(ccc1(1,loop+1),ccc2(1,loop+1),5),':');
end
plot(xrange,(ccc(1,1)*yrange),':');
title('X3_Band')
text(-19,.8,': =X2_X3#db')
xlabel('dB')
ylabel('Corr. Coef.')
hold off
pause

clg
%Bhattacharyya Distance %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PLOT#1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%UR
subplot(221)
cla
plot(range,eee(1,db_len),':');
axis([-20 0 0 1])

hold on
plot(xrange,(eee(1,1)*yrange),':');

plot(range,eeee(1,db_len),'-');
plot(xrange,(eeee(1,1)*yrange),'-');

plot(range,e(2,db_len),'--');
plot(xrange,(e(2,1)*yrange),'--');

text(-19,.9,': =X2_ X3#db')
text(-19,.8,'_ =X2_X1#db')
text(-19,.7,'--=X2_X2#db')
%text(-14,.8,'Straight line = no_noise level')
title('Bhattacharyya Distance')
xlabel('dB')
ylabel('Distance')
hold off

%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)
cla
```

```
plot(range,eeee(1,db_len),'-');
axis([-20 0 0 1])
hold on
plot(xrange,(eeee(1,1)*yrange),'-');
for loop=1:9;
    plot(band*range1(loop),linspace(eeee1(1,loop+1),eeee2(1,loop+1),5),'-');
end
title('X1_Band')
text(-19,.8,': =X2_X1#db')
xlabel('dB')
ylabel('Distance')
hold off
%PLOT#3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(223)
cla
plot(range,e(2,db_len),'--');
axis([-20 0 0 1])

hold on;
plot(xrange,(e(2,1)*yrange),'--');
for loop=1:9;
    plot(band*range1(loop),linspace(e1(2,loop+1),e2(2,loop+1),5),'--');
end
title('X2_Band')
text(-19,.8,': =X2_X2#db')
xlabel('dB')
ylabel('Distance')
hold off
%PLOT#4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(224)
cla
plot(range,eee(1,db_len),':');
axis([-20 0 0 1])

hold on
for loop=1:9;
    plot(band*range1(loop),linspace(eee1(1,loop+1),eee2(1,loop+1),5),':');
end
plot(xrange,(eee(1,1)*yrange),':');
title('X3_Band')
text(-19,.8,': =X2_X3#db')
xlabel('dB')
ylabel('Distance')
hold off

pause

clg
% Divergence
%PLOT#1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%UR
subplot(221)
cla
plot(range,ddd(1,db_len),':');
```

```
axis([-20 0 0 20])

hold on
plot(xrange,(ddd(1,1)*yrange),':');

plot(range,dddd(1,db_len),'-');
plot(xrange,(dddd(1,1)*yrange),'-');

plot(range,d(2,db_len),'--');
plot(xrange,(d(2,1)*yrange),'--');

text(-12,17,': =X2_ X3#db')
text(-12,14,'_ =X2_X1#db')
text(-12,11,'--=X2_X2#db')
%text(-14,.8,'Straight line = no_noise level')
title('Divergence')
xlabel('dB')
ylabel('Divergence')
hold off
%PLOT#2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(222)
cla
plot(range,dddd(1,db_len),'-');
axis([-20 0 0 20])

hold on
plot(xrange,(dddd(1,1)*yrange),'-');
for loop=1:9;
    plot(band*range1(loop),linspace(dddd1(1,loop+1),dddd2(1,loop+1),5),'-');
end
title('X1_Band')
text(-12,17,': =X2_X1#db')
xlabel('dB')
ylabel('Divergence')
hold off
%%%%%%%%%%%%%%
%PLOT#3
%%%%%%%%%%%%%%
subplot(223)
cla

plot(range,d(2,db_len),'--');
axis([-20 0 0 20])
hold on;
plot(xrange,(d(2,1)*yrange),'--');
for loop=1:9;
    plot(band*range1(loop),linspace(d1(2,loop+1),d2(2,loop+1),5),'--');
end
title('X2_Band')
text(-12,17,': =X2_X2#db')
xlabel('dB')
ylabel('Divergence')
hold off
%%%%%%%%%%%%%%%%
%PLOT#4 %%%%%%%%%%%
```

```
subplot(224)
cla

plot(range,ddd(1,db_len),':');
axis([-20 0 0 20])
hold on
for loop=1:9;
    plot(band*range1(loop),linspace(ddd1(1,loop+1),ddd2(1,loop+1),5),':');
end
plot(xrange,(eee(1,1)*yrange),':');
title('X3_Band')
text(-12,17,': =X2_X3#db')
xlabel('dB')
ylabel('Divergence')
%print -deps test12
hold off
```

C. NEURAL NETWORK INPUT PROGRAMS

C.15: Program NN_Setup.M [8]

```
%*************************************************************
%      NEURAL NETWORK INPUT DATA SET-UP
%*************************************************************

%   version 1.2  2/2/94
%(modified from the original program completed by Dr. Monique P. Fargues)
%   set up the fourier coefficients for nn inputs
%   6 coeffs/line (input) 2 outputs; set-up for 50 coefs
count=input('How many times do you want to run this program »  ');
for loop4=1:count
clear X x
Q=[ ' input coef. matrX3_no_noiseix x(nlines,ncol)       '
   ' nlines: number of fft coefs used '
   ' ncol:   number of pulses used'];
disp(Q)

FC=input('input nn coefs. matrix »  ','s');    % x(nlines,ncol)
X=eval(FC);
X=X';  % This takes the transpose
mat=[FC '.nna'];
% nlines: nb of coefs
% ncol:  nb of pulses
% investigated
%Ncol=input('number of fft coefs to be kept (ie. most info. in 30) »  ');
Ncol=30;

%Q2=[ 'choose between 2 or 3 different sets to train the NN on'];
Q2=[ 'This program will use 2 different sets to train the NN on'];
disp(Q2)
```

```
%iset=input('Number of classes » ');
iset=2;

%iopt=input('Class of the pulse: 1(for X1#)/2(for X2#)/3 ');
iopt=input('Class of the pulse: 1(for X1#) or 2(for X2#) or 3(for X3#) » ');
%idb=input('Use log coefs y/n:1/0 » ')
idb=1;

x=X(1:Ncol,:);      % restrict to the desired nb of fft coefs
[nlines,ncol]=size(x);
Npts=nlines;
N_lines=fix(Npts/6) ;           % 6 sampes/line
N_end=Npts-N_lines*6;              % last line
% amplitude normalization of the pulses need for NN
% normalize power in X pulses to 1, per pulse.

if idb==1,x=10*log(x);end       % choice to use coefs in dB

 for k_puls=1:ncol
 temp=sum(x(:,k_puls));
   x(:,k_puls)=x(:,k_puls)/temp;
 end

% set up for bad or good pulse identification
if iset==2                    % test for 2 classes
  if N_end==0,
    if iopt==1, ref=['&  1   0',13];, else, ref=['&  0   1',13];end
  else
    if iopt==1, ref=['  1   0',13];, else, ref=['  0   1',13];end
  end
else                         % test for 3 classes
  if N_end==0,
    if iopt==1,     ref=['&  1 0 0',13];
    elseif iopt==2, ref=['&  0 1 0',13];
    elseif iopt==3, ref=['&  0 0 1',13]; end
  else
    if iopt==1,     ref=['  1 0 0',13];
    elseif iopt==2, ref=['  0 1 0',13];
    elseif iopt==3, ref=['  0 0 1',13]; end
  end
end
%  output data per pulse
for kp=1:ncol
% first line
fprintf(mat,'%11.8f %11.8f %11.8f',x(1,kp),x(2,kp),x(3,kp))
fprintf(mat,' %11.8f %11.8f %11.8f\n',x(4),x(5,kp),x(6,kp))

for kl=1:N_lines-1
fprintf(mat,'&   %11.8f %11.8f %11.8f',x(6*kl+1,kp),x(6*kl+2,kp),x(6*kl+3,kp))
fprintf(mat,' %11.8f %11.8f %11.8f\n',x(6*kl+4,kp),x(6*kl+5,kp),x(6*kl+6,kp))
end
if N_end ~=0                  % complete for the last line
  kl=N_lines
 if N_end >= 1, fprintf(mat,'&   %11.8f',x(6*kl+1,kp));end
 if N_end >= 2, fprintf(mat,' %11.8f',x(6*kl+2,kp));end
 if N_end >= 3, fprintf(mat,' %11.8f',x(6*kl+3,kp));end
 if N_end >= 4, fprintf(mat,' %11.8f',x(6*kl+4,kp));end
 if N_end >= 5, fprintf(mat,' %11.8f',x(6*kl+5,kp));end
end
fprintf(mat,ref),        % enter output parameters
end
clear idb iopt iset k_puls kl kp mat ncol nlines ref temp x X
clear FC N_end N_lines Ncol Npts Q Q2
end
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
4.	Professor Ralph Hippenstiel, Code EC/Hi Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	2
5.	Professor Monique P. Fargues, Code EC/Fa Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	2
6.	David A. DeRieux, Code 8133 Space Systems Development Department Naval Research Laboratory 4555 Overlook Ave. Washington, D.C. 20375-5000	2